Process-Tolerant VLSI Neural Networks for Applications in Optimisation

Donald J. Baxter

Thesis submitted for the degree of
Doctor of Philosophy
The University of Edinburgh
January 1993

Acknowledgements

I would like to acknowledge the help and support I have received from many people during the time taken to complete the research for this thesis.

- I would like to thank both of my supervisors, Alan Murray and Martin Reekie, for their help, advice and support over the last 3 years.
- Special thanks are due to Alister Hamilton and Stephen Churcher without whose encouragement and help the work contained within this thesis would not be as complete.
- Thanks are due to the Science and Engineering Research Council for their financial support and Thorn EMI's Central Research Laboratories for in Hayes for CASE sponsorship. In particular at Thorn EMI I want to thank Chris Sharpington, Neil Burgess for their invaluable advice about neural optimisation techniques.
- Finally I would like to thank Robin Woodburn and Anne Moore for their patience and diligence in proof reading this thesis.

Declaration

The work presented in this thesis was carried out by the author, unless otherwise stated.

Donald J. Baxter

Abstract

Optimisation problems such as scheduling and resource allocation are hard, as large numbers of solutions exist for "real" problems. Neural networks have been reported to find optimal solutions quickly. These networks derive their power from a massively parallel architecture, drawing its inspiration from the biological nervous system.

There is a need for dedicated hardware implementations to accelerate neural computations. There is also a desire to develop autonomous neural systems.

Digital circuits are tolerant of process variations. Digital circuits are however large. Analogue circuits are more compact and consume less power, but are dependent on the fabrication process for correct functionality. The choice between the two techniques is determined ultimately by the application. Analogue techniques are necessary to obtain a completely parallel implementation. Both the Hopfield/Tank and Kohonen networks used in optimisation rely upon matched circuit elements. Thus the development of process invariant analogue circuits for neural networks was the major aim of this thesis.

Simulations are reported, of the Hopfield/Tank and the Kohonen networks, applied to the 10-city Travelling Salesman Problem (TSP). They confirm the reported optimisation abilities. The Kohonen network is shown to be faster and more robust than the Hopfield/Tank network.

The results obtained from two fabricated devices are reported: a small scale test-chip and a large scale, generic building block device (EPSILON). These results show that the circuits developed in this thesis offer a significant immunity to the effects of process variations. The Kohonen network for the TSP was implemented on EPSILON. The Kohonen network was a very tough test for EPSILON, in that it requires a high degree of accuracy to be able to discriminate between the responses of neurons. Process variations prevented EPSILON from solving TSPs greater than 9 cities.

The main conclusion of this thesis is that unless the neural algorithm actively compensates for the effects of process variations, the performance of a network implemented in analogue VLSI is compromised.

Table of Contents

Chapter 1

Introduction

The work in this thesis combines a personal interest by the author in VLSI design with a desire by Thorn-EMI's Central Research Laboratories for a fast, high performance optimisation system. In a company such as Thorn-EMI, the quest for efficiency is a never-ending one. The company is always searching for a new competitive edge. Consequently there is a continuous drive towards developing better techniques for organising schedules or resources, to reduce both the time and the cost of projects.

The work in the field of optimisation spans more than half a century and has it roots in mathematics and operational research. Early optimisation techniques were based on differential calculus or linear programming. The increase in computing power over the past few decades has enabled the development and implementation of sophisticated search techniques for solving a wide variety of optimisation problems. Neural networks are comparatively recent arrivals in the field of optimisation. Their inspiration is the massively parallel nature of the human nervous system, which despite the slow response of the individual components (neurons) is able to solve optimisation problems quickly due to the high level of in-built parallelism. Papers by Hopfield/Tank [1] and Angeniol [2] have reported results which suggest that neural networks possess the much-prized ability to find optimal or near-optimal solutions to the problem in hand quickly.

Unfortunately, due to the Von Neumann architecture of conventional computers, simulations of a neural networks serialise the computation, losing the speed advantage of the completely parallel computation in a neural network. To speed up the simulations of neural networks, and with a view to the development of autonomous neural systems, a wide variety of dedicated hardware implementations have been developed. The approaches include the use of high performance specialist processors, or custom VLSI devices optimised specifically for neural networks. In both cases the implementation normally possesses a degree of parallelism. Custom VLSI implementations divide into 2 generic sections:

1 Digital Techniques

2 Analogue Techniques

The extensive use of digital design techniques within conventional computers means that digital technology is more familiar. As a result, general purpose neural accelerator cards normally use digital circuits. This also makes interfacing to a digital host computer

straightforward. The strengths of analogue circuits are, however, small size and low power consumption. Thus they are well suited for stand-alone implementations where a large network has to be integrated as a small number of devices. This demonstrates not that one technique is inherently **better-suited** to implementing VLSI neural networks, but that **the choice is determined by the requirements of the proposed application of the neural VLSI implementation**.

The Hopfield/Tank neural network is derived from a simplified analogue model of biological neurons. Digital implementations are thus inevitably only an approximation to a network's true response. As the digital implementation has to simulate the characteristics of the analogue elements which make up the Hopfield/Tank network, an analogue implementation **should** be more accurate, and also faster. This, coupled with the relatively large size of the Hopfield/Tank neural network, means that here an analogue implementation is appropriate. As the Kohonen neural network has a similar basic architecture to the Hopfield/Tank network, the same VLSI implementation is suitable for both networks.

The tolerances of the fabrication process for an integrated circuit mean that the characteristics of the process will vary between chips and to a lesser extent within a single chip. The **degree** of the resultant mis-match between the performance of transistors depends on the quality of the fabrication process. The function of digital circuits is relative insensitive to these problems as transistors are either OFF or ON. Unfortunately, as the compactness of analogue circuits is gained through the exploitation of the transfer characteristics of a transistor, the overall response of the circuit is heavily dependent on the fabrication process. Thus two otherwise identical circuits may perform differently. Both the Hopfield/Tank and the Kohonen neural network rely on the elements within the network having exactly the same performance. When this is not the case the optimisation abilities of the network are compromised.

As a result, the development of process invariant analogue circuits for neural networks became a central pillar of this thesis. Tackling the problem at the circuit level rather than an algorithmic level yields a solution which extends the range of applications for analogue VLSI in general and for neural networks in particular.

The organisation of the remainder of this thesis is as follows. Chapter 2 introduces conventional and neural optimisation techniques. The characteristics of each technique are discussed. The classic optimisation problem the Travelling Salesman Problem (TSP) is also described. Simulation results are presented for a three way comparison of the exhaustive search technique, and the Hopfield/Tank and the Kohonen neural networks, all applied to the 10 city TSP. The need for a dedicated VLSI implementation is identified.

Chapter 3 reviews the digital and analogue VLSI design techniques which have been used to implement neural networks. Chapter 4 describes the advantages of using a hybrid digital/analogue circuit based on encoding the neural state value as a stream of pulses. The chapter then outlines the development of a process invariant neural system based on pulse encoding.

Chapter 5 reports the results from the small neural test array fabricated to characterise the process invariance of the circuits developed in Chapter 4. The successes and limitations are identified and an improved set of circuits is proposed.

The modifications to the circuits based on the experiences of the testing of the test-chip are described at the start of Chapter 6, where the specifications for the resultant generic neural building block device and its support system are also discussed. Characterisation results for the device are presented along with a simple hardware/software comparison.

Chapter 7 brings together all the software and hardware results to draw an overall conclusion about the performance of the Kohonen network implemented on analogue hardware. Areas for future work are outlined.

Chapter 2

Optimisation Techniques

The field of optimisation techniques and systems is both wide and diverse, ranging from highly analytical techniques like differential calculus through to intelligent search algorithms based on heuristics. This breadth results from 2 factors:

1 Optimisation is an everyday occurrence. What is the shortest and quickest route from shop A to shop B ? What is the best order to cook a meal so that all dishes are ready at the same time? Clearly the range of optimisation applications is very wide.

2 In industry, the competitiveness between companies means that there is a constant drive to improve efficiency and throughput through more efficient scheduling and better assignment of resources. A change amounting to as little as a few percent can make the difference between profit and loss.

Most of the problems contained in the literature relate to the industrial environment. The task assignment and job-shop scheduling problems are the most commonly quoted examples. Both of these problems require a limited set of resources to be organised so that either the total cost/time is minimised or the throughput is maximised. Mathematically this can be thought of as a cost function, the value of which has to be minimised/maximised subject to the constraints imposed by limited resources. This class of problem is thus termed constrained optimisation.

Working out the optimal task assignment or schedule for a problem is a computationally intensive task owing to the large number of possible assignments and schedules that exist for a realistically sized problem. The limitations placed by the constraints complicate matters further. To illustrate the computational complexity of such problems, let us take the example of scheduling N tasks on a single machine. Each task has a "setup cost" associated with it which depends on the task which precedes it. For this problem there are N! possible schedules. Thus the computational time required to carry out an exhaustive search of all the schedules will increase exponentially with the problem size. Such problems are termed non-deterministic polynomial-complete (NP-complete). Thus to solve optimisation problems a better approach is needed than the brute force technique of an exhaustive search.

The more intelligent approaches fall into 2 main categories:

1 Traditional methods - Graph theory, Linear/non-linear programming, Heuristics and Simulated Annealing.

2 Neural Networks - Hopfield/Tank, Boltzmann Machines, Stochastic and Kohonen.

The above division can become blurred as neural techniques borrow many ideas from the longer established conventional optimisation methods. For example Stochastic Neural Networks and Boltzmann Machines use noise sources in a similar way to the process of simulated annealing to reduce the likelihood of finding a sub-optimal solution.

The main difference between traditional and neural approaches is that, while in the traditional approach the algorithms are fundamentally serial, in a neural network the computation is at least structurally parallel. In a loose sense the network "tries out" all possible solutions in parallel, and the best solution wins as the system converges to a stable state. Due to the serial computation of a conventional Von Neuman computer the parallel nature of a neural network effectively becomes serialised. Thus developing a truly parallel neural network implementation will greatly improve the speed of a neural optimisation system.

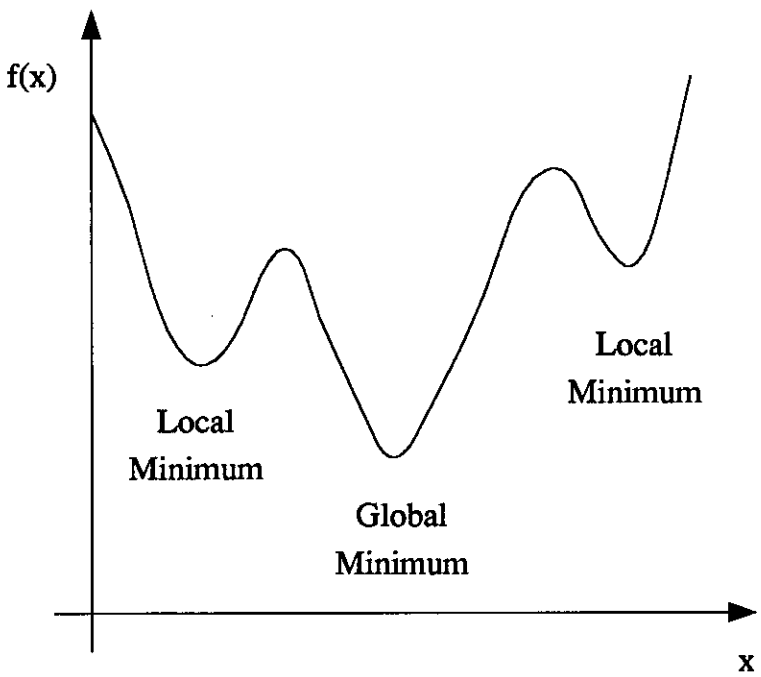


Figure 2.1 Local and Global Minima.

2.1. Local and Global Minima

In most optimisation problems many solutions exist which satisfy the specified constraints. However the **cost** of each solution will not necessarily be the same. Normally only one solution gives the lowest cost (the global minimum) while the remainder are only locally optimal (local minima). Figure 2.1 shows local minima and the global minimum for a simple cost function.

As many optimisation techniques act to reduce the value of the cost function by following the gradient of the cost function until it reduces to zero, it is common for an algorithm to find a local rather than the global minimum. While more complex algorithms will increase the probability of finding the global rather than a local minimum the computational load required is also greater. In general a trade is made between the computation time and the certainty bounds on the solution quality. Fortunately, all that is often wanted is a good solution to the problem in hand rather than the optimal solution. As a result the algorithms developed for optimisation problems tend to concentrate on producing an answer very quickly, on the basis that the algorithm can be run several times from different start points to increase the probability that a good solution is found.

2.2. The Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is probably one of the best known optimisation problems, simply because it is easy to describe but very hard to solve optimally.

The problem faced by the salesman is that he must visit all N cities on his list once, returning to his starting point, while minimising the total distance travelled. Figure 2.2 contains an example of a 10-city TSP. Superficially solving such a problem does not seem hard. The large number of possible tours however renders it a hard problem to solve algorithmically.

For N cities there are N! possible tours. As the salesman must return to his starting point, a complete tour is circular. Thus the length of the tour is not affected by the city in which the tour starts. This reduces the number of different tour lengths by a factor of N to (N – 1)! The circular nature of the tours also means that the direction of the tour does not affect the length of the tour, reducing the number of tours by a factor of two. This gives the number of distinct tours for the N city TSP as:-

$$\frac{(N-1)!}{2} \tag{2.1}$$

Equation 2.1 clearly demonstrates that for an exhaustive search of all the possible tours the computation time increases exponentially with the problem size. Thus the TSP belongs to the NP-complete class of problems.

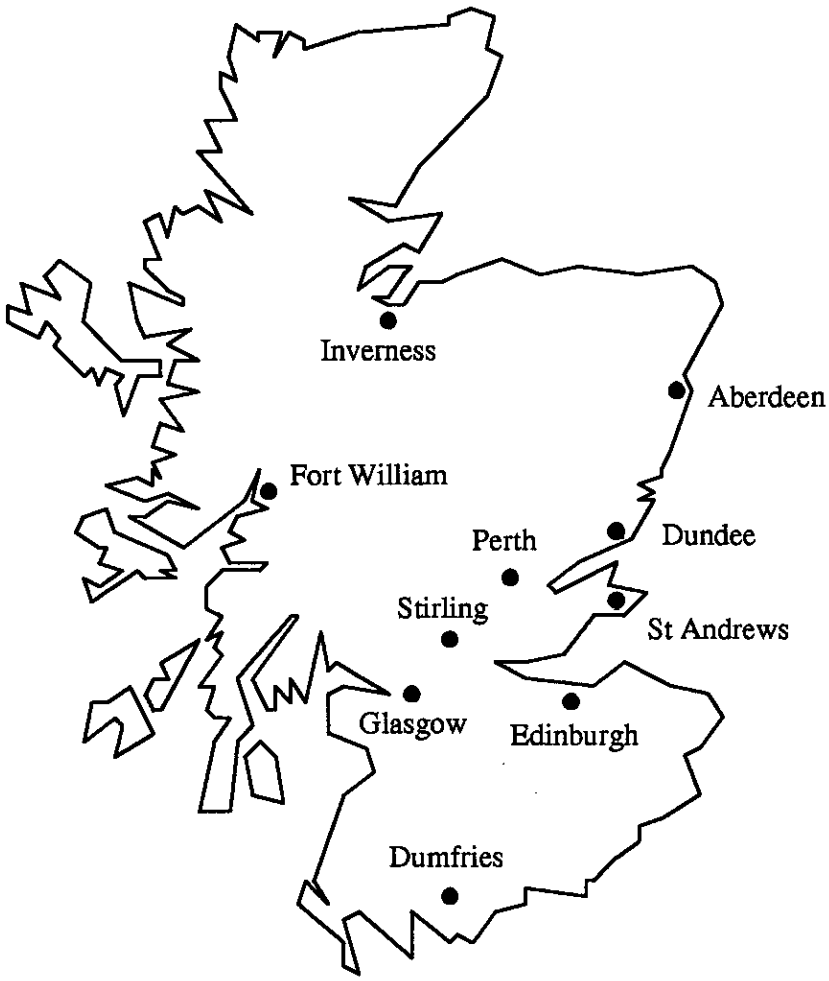


Figure 2.2 The Travelling Salesman Problem.

In practice the TSP has become the benchmark through which researchers demonstrate the effectiveness of their algorithms. While this may allow easy comparison of the algorithms, good performance on the TSP does not guarantee that the algorithm will perform well on other problems. Many scheduling algorithms are based on heuristics which are only applicable to the problem in hand and thus are not well suited to other problems. Such "tweaking" is often necessary in order to boost the performance of an algorithm.

The remainder of this chapter reviews both conventional and neural techniques used to solve optimisation problems and the results from a three way performance comparison of the exhaustive search method, the Hopfield/Tank Neural Network and the Kohonen Self-Organising Neural Network applied to the TSP.

It should be emphasised that this section is not a comprehensive review of optimisation techniques but rather is aimed at putting into context the neural optimisation demonstrator running on the circuits described in Chapters 4, 5 and 6.

2.3. Conventional Optimisation Techniques

For simple cost functions, calculus can be used to find the positions of the maxima and minima. However, for complex cost functions finding the zeroes of the first derivative of the cost function is often difficult. In many optimisation problems the set of valid solutions is a discrete data set resulting in discontinuities in the cost function. As a result of these problems, cost functions are normally minimised by iterative search methods like steepest gradient descent. In this case the algorithm moves down the line of steepest gradient from the chosen start point. When the gradient reduces to zero a minimum has been found. Unfortunately there is no guarantee that this minimum is the global minimum of the function and not a local minimum. Normally, iterative techniques such as gradient descent are run from many different start points to increase the probability of finding a good, or perhaps the optimal, solution.

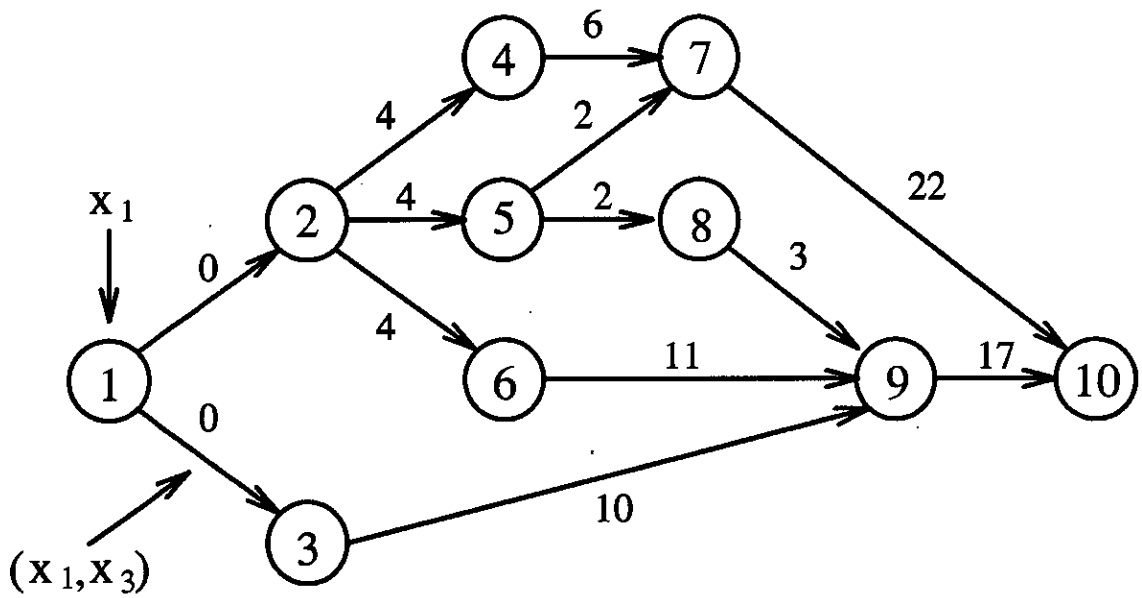


Figure 2.3 An Example of a Graph/Network.

2.3.1. Graph Theory and Network Flows

The first step in solving an optimisation problem is to specify the problem so that either a matrix equation or a system of equations can be formulated. Graph theory and network flows are examples of such techniques. The remainder of this section outlines these optimisation techniques.

A n-node graph G = (X, U) is defined by two data sets

1 **Vertex** (Node) Set, $X=\{x_1, x_2, \ldots, x_n\}$

2 **Edge** (Arc) Set, $U=\{(x_1, x_2), \ldots\}$

where U is a subset of all the possible links between the set of **vertices**, X.

A **vertex** represents either a machine or a city (TSP). The links between the **vertices** (**edges**) define the cost of moving from one **vertex** to another. In flow networks each **edge** has a maximum flow limit and the present flow value associated with it.

Each **edge** can have a label associated with it and usually represents the cost of moving from node i to node j. As Figure 2.3 shows, this graph notation is a very natural way to describe scheduling problems. From these sets an $n \times n$ adjacency matrix and an incidence matrix can be derived; these are used to create a system of equations from which the problem defined by the graph can be solved.

The method of solving these equations is highly problem-specific and depends on the type of matrix produced by the problem. Some graphs give matrices which are either upper or lower triangular allowing the system to solved simply by forward or backward substitution. Other graphs may give systems of equations which can only be solved by iterative techniques. The TSP yields a matrix which is symmetric about its leading diagonal. It is also an example of a Hamiltonian cycle, as a solution traverses every node of the graph exactly once.

Networks are similar to graphs except that the label represents the maximum flow capacity for that **edge**. There is a start node (source) and an end node (sink). The problem, given the various constraints, is normally to ask: what is the maximum flow which can flow between the source and the sink?

The flow conservation rule for networks states that all flows at a particular node must sum to zero (equivalent to Kirchhoff's Current Law in circuits).

The Ford-Fulkerson method [3] is one of the classic algorithms for determining the maximum flow across a network. It is based on 3 ideas:

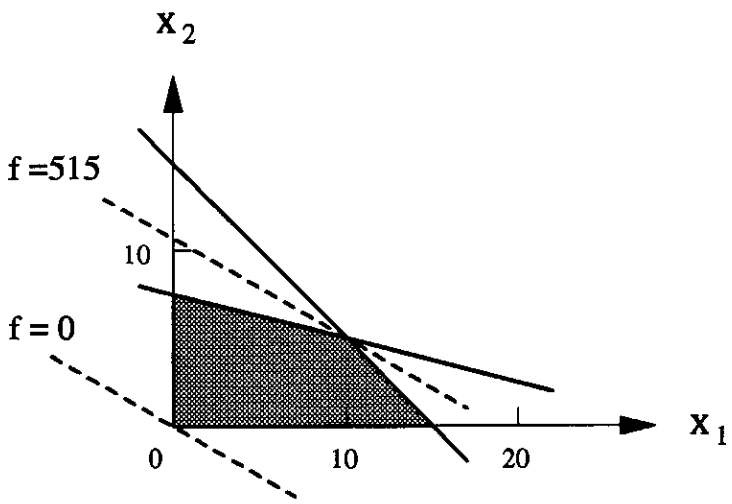
1 Residual networks.

2 Augmenting paths.

3 Max-flow-min-cut theorem.

Residual networks are simply graphs of the differences between the capacity of the edges in the network and their actual flow. An augmenting path of a residual network is a path running from the source to sink nodes in which the flow can be increased to augment the overall flow. The Ford-Fulkerson procedure looks for an augmenting path which is the shortest residual path in the residual network. Then the flows in all the edges in this path are incremented by the value of the smallest residual capacity on this path. This cycle is repeated until no more augmenting paths can be found. At this point the max-flow-min-cut theorem states that the flow through the network is at a maximum.

The speed of this procedure rests on how quickly the paths containing the smallest residual can be identified.



Cost Function:

$$f = 29x_1 + 45x_2$$

Constraints:

$$2x_1 + 8x_2 \leqslant 60$$

$$4x_1 + 4x_2 \leqslant 60$$

$$x_1 \geqslant 0$$

$$x_2 \geqslant 0$$

Figure 2.4 A Simple Linear Programming Example.

2.3.2. Linear and Non-Linear Programming

In linear programming a cost function f is optimised subject to a list of constraints on the values of the variables in the cost function, i.e. maximise

$$f = c_1x_1 + c_2x_2 + \cdots c_nx_n \tag{2.2}$$

subject to

$$\begin{aligned} &a_{11}x_1 + \cdots + a_{1n}X_n = b_1 \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ &a_{m1}x_1 + \cdots + a_{mn}X_n = b_m \\ &x_i \geq 0 \quad (i = 1, \ldots, n) \end{aligned} \tag{2.3}$$

where m is the number of constraint equations and n is the number of variables. If m=n then a unique solution for the problem should exist. It is more common, however, for m<n, which gives a solution region rather than a single point. The shaded area in Figure 2.4 shows the solution region for a simple example. The maximum value of the cost function lies at one extreme point or vertex of this region. A vertex is defined as the point where any (n – m) of the variables are zero.

One of the main techniques for solving linear programming problems is the Simplex search method developed by Dantzig [4]. The basic operation behind this method is to move from the current vertex to whichever neighbouring vertex will increase the value of the cost function by the greatest amount. The step is then repeated until the value of the

cost function can no longer be increased.

Equations 2.4 and 2.5 show the N-city TSP expressed as a linear programming problem

$$f = \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} x_{ij} \tag{2.4}$$

subject to

$$\begin{aligned} &(a) \quad x_{ii} = 0 \\ &(b) \quad x_{ij} = 0, 1 \\ &(c) \quad \sum_{i=1}^{N} x_{ij} = \sum_{j=1}^{N} x_{ij} = 1 \end{aligned} \tag{2.5}$$

where $1 < i, j < N$. The main problem experienced by Dantzig *et al*[5] when they optimally solved a 42-city TSP using linear programming was the large number of constraints required. To reduce this overhead the symmetric nature of the TSP distance matrix was exploited.

In non-linear programming both the cost function and the constraints may be non-linear i.e. they may contain powers of x_i. As the shape of the solution region is now complex, this class of problem is much harder to solve. Some techniques simplify the problem by approximating the non-linear constraints by several linear constraints. In general, solutions are obtained using iterative search techniques like the Newton-Lagrange method. A discussion about methods of solving non-linear programming is not appropriate in this general overview of optimisation techniques. A good description of non-linear techniques and the mathematics behind them is given in [6].

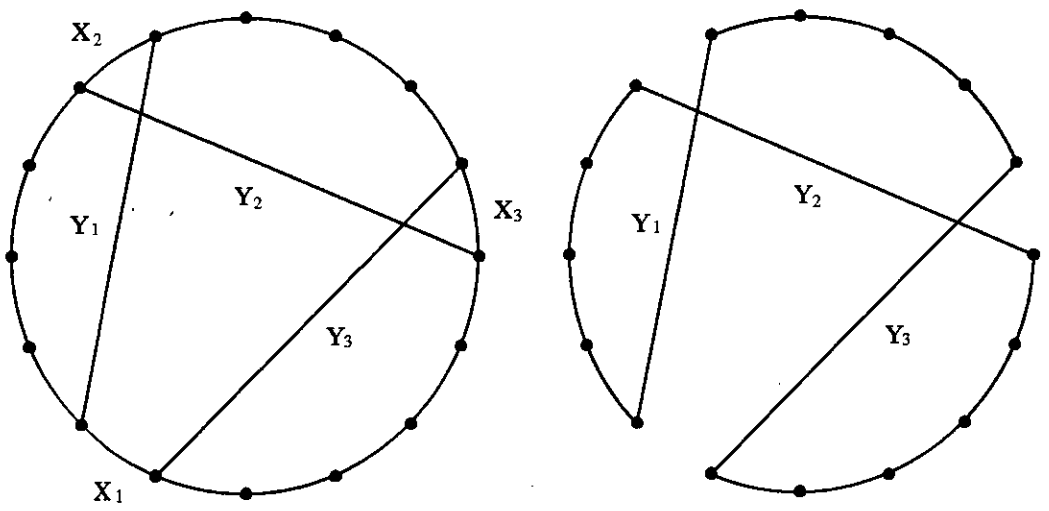


Figure 2.5 Sequential Exchange Search Method.

2.3.3. Heuristics

Heuristics are guidelines which influence, rather than specify the direction of the search for the optimal solution. These guidelines can either exploit particular facets of a specific problem or may define a general search technique. The Lin and Kernighan algorithm [7-9] is probably the best known heuristic method for solving the TSP.

The basic sequence of operations is the same as in most iterative search techniques.

1 First choose a (pseudo) random start point.

2 Apply a transformation to the present tour in an attempt to improve it.

3 If the new tour is better then keep tour and repeat from 2, otherwise reject.

4 If an improved tour cannot be found then tour is locally optimal. Repeat from 1 until either computation time runs out or answer is satisfactory.

Genetic Algorithms [10] are based on the process of evolution of biological species and as the basic mutation/selection cycle below shows, they are functionally similar to the heuristic search procedure outlined above:

1 Carry out a random alteration (mutation) to the trial solution.

2 Keep the mutation if it lowers the cost (improves the fitness) of the solution, otherwise keep old solution.

3 Repeat 1 and 2 until no further improvement is possible.

In the Lin and Kernighan algorithm the method of tour to tour improvement is based on a link (edge) exchange strategy (Figure 2.5). A connected sequence of k links is replaced by k different links. If this exchange reduces the tour length then it is kept, otherwise it is discarded. The exchange process carries on until swapping k links yields no further reduction in tour length. At this point the tour is locally optimal. The k links in the tour to be replaced are selected by identifying exchanges which will potentially give the greatest reduction in the length of that sequence.

To enhance the performance of this exchange process, and to reduce the computational load, a range of other techniques is employed. The first is back tracking to allow the algorithm to escape from tour sets which may be locally optimal but are globally poor. The level of backtracking is limited to 2 otherwise the computation time increases greatly.

Once a locally optimal tour has been found, time is required to confirm its optimality by checking other possible exchanges. To reduce this "checkout time" a note of the tours found so far is kept so that when a tour is found for a second time the algorithm stops and then restarts from a new start point.

As the decision to make or break links is made on a "nearest neighbour first" basis, only local information is used. To get around this, limited lookahead is employed so that

more is known about the consequences of the proposed swap on the overall tour length.

The locally optimal tours produced will tend to have links in common. The algorithm recognises these "good" links in the tours produced so far and then fixes them for subsequent runs. This can be viewed as directing the search based on information gained from previous runs.

The final heuristic is to apply non-sequential exchanges to the links which are not common to other tours to try to convert what may be a non-optimal into an optimal tour.

The results reported show that for the 100-city TSP a single run took 25s (Fortran A on a GE635). To find the optimal tour with a 95% certainty the algorithm had to be run repeatly from different start points for 3 minutes. As computing power has advanced considerably since these results were reported in 1973, it is not unreasonable to expect the Lin and Kernighan algorithm to run at least 3 or 4 orders of magnitude faster on present day computers.

2.3.4. Simulated Annealing

As has already between mentioned, the main problem with iterative search techniques based on gradient descent is that there is no guarantee that the minimum found is global. The Lin and Kernighan algorithm used heuristics to increase the probability of finding the global minimum. Kirkpatrick *et al* [11, 12] use a different method based on statistical mechanics to avoid local minima and find the optimal solution with a high probability.

The idea stems from the growing of a single crystal from a molten melt. If the substance is cooled (annealed) properly then a single crystal will form. However if the cooling schedule deviates from the desired path the resulting crystal will have many defects or a glass may be formed. A single crystal is the lowest energy state form of the substance; other forms are locally optimal states into which the substance is trapped if the cooling is too rapid.

The work by Kirkpatrick *et al* is based on the Metropolis procedure for modeling the thermal motion of atoms, at a given temperature, T. The main steps of the Metropolis algorithm are shown below.

1 Give an atom a small random displacement.

2 Calculate change in energy, ΔE, for system.

3 If $\Delta E \leq 0$ then keep new configuration. Otherwise compare a random number between 0 and 1 with the probability $P(\Delta E) = \exp(-\Delta E/kT)$. If the random number is less than $P(\Delta E)$ then keep, if not reject.

4 Repeat from 1.

To convert the above procedure into an optimisation technique Kirkpatrick *et al* replaced the energy function for the atoms with a cost function and defined the possible configurations by a set of parameters $\{x_i\}$. Optimisation is achieved by cooling the system from a high initial temperature to its freezing point. The initial high temperature allows the system to make a large number of uphill moves. As the system cools, uphill moves become less likely. It is this hill climbing ability which gives simulated annealing the ability to escape from local minima. Thus by the time the temperature of the system reaches freezing point, the system should have found the global minimum.

A simple way of viewing this is to think of the temperature controlling the magnitude of a noise source. At high temperatures the high level of noise allows the system to "jump around" exploring large areas of the cost function. As the level of the noise is reduced the system becomes trapped in the deepest region of the cost function out of which it is not able to jump, so progressively homing in on the global minimum.

By formulating the appropriate cost functions, Kirkpatrick *et al* have used this technique to solve successfully a wide range of optimisation problems, for example:

1 The TSP.

2 VLSI cell placement.

3 The routing of VLSI tracks.

4 The partitioning of function blocks between integrated circuits.

While this method generally yields solutions of high quality, the very slow nature of cooling required to give high quality solutions results in long computation times.

2.4. Neural Techniques

Interest in artificial neural networks for optimisation was stimulated by the seminal paper by Hopfield/Tank[1]. Although the basic unit of this structure, the neuron, is functionally simple and has a slow time constant, the combination of many units working in massive parallelism yields a powerful processing system. Information in a neural network is encoded in the strengths of the synapses which inter-link the neurons. The distributed representation makes neural networks tolerant to effects of faults in either synapses or neurons. Thus neural networks are well suited to implementation in VLSI hardware where processing defects may cause synapses and neurons to fail.

A more formal description of the operation of a neural network is as follows. The common factor in all neural network models is a column of N synapses feeding into a neuron. As Equation 2.6 shows the activity value for neuron i, x_i, produced by the synaptic column, is the scalar (dot) product of the neural state input vector $\{V_0,..,V_j,..,V_N\}$ and

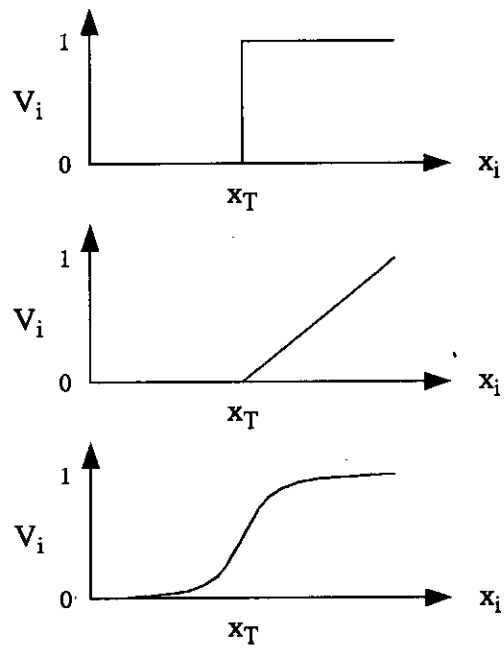
the vector represented by the synaptic weights $\{V_{Ti0},..,V_{Tij},..,V_{TiN}\}$.

$$x_i = \sum_{j=0}^{N-1} V_{Tij} V_j \tag{2.6}$$

This activity is "compressed" into a neural state, V_j, by the neuron applying a monotonically increasing activation function $f(x_i)$.

$$V_j = f(x_i) \tag{2.7}$$

Figure 2.6 illustrates some of the forms this activation can take. The neuron's threshold value, x_T, sets the point around which this activation varies. For a complete neural network, Equations 2.6 and 2.7 translate into an array of multipliers which perform vector-matrix multiplication and a row of cells with variable input-output characteristics which convert the activation values into the required neural state.



Threshold Activation Function

$f(x_i) = 0$ if $x_i < x_T$

$f(x_i) = 1$ if $x_i > x_T$

Linear Activation Function

$f(x_i) = 0$ if $x_i < x_T$

$f(x_i) = x_i - x_T$ if $x_i > x_T$

Sigmoid (S-shaped) Activation Function

$$\text{e.g. } f(x_i) = \frac{1}{1 + \exp(-(x_i + x_T)/T)}$$

Figure 2.6 Activation Functions.

At present four different neural network models have been applied to a wide variety of optimisation problems.

1 Hopfield/Tank Neural Network

2 Stochastic Neural Network

3 Boltzmann Machines

4 Kohonen Self-Organising Neural Network

The remainder of this section discusses the pertinent features of these networks in more detail.

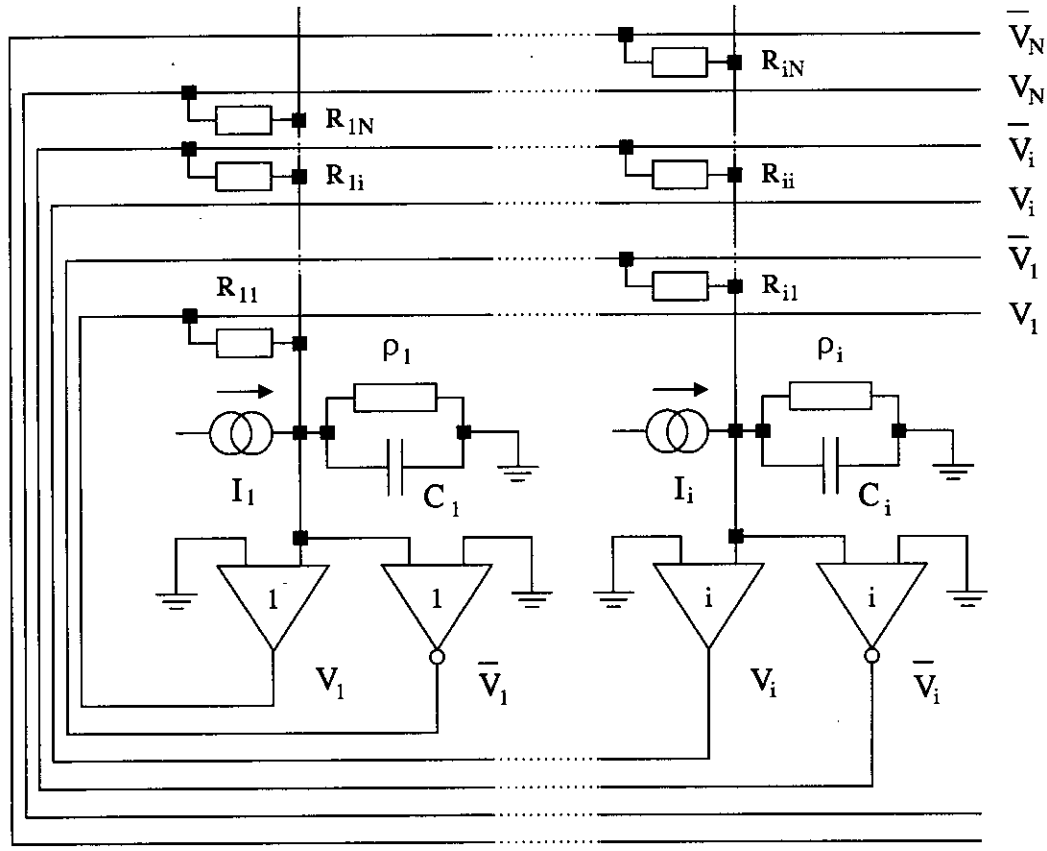


Figure 2.7 The Architecture of the Hopfield/Tank Neural Network.

2.4.1. The Hopfield/Tank Neural Network

The Hopfield/Tank Neural Network [13, 14, 1] is a fully interconnected network (N neurons, N^2 synapses). The synapses take the form of resistors feeding into an operational amplifier (the neuron) which applies a non-linearity to the sum of the input currents (Figure 2.7). These input currents are summed using a parallel RC network between the input to the amplifier and ground. The capacitor gives the neuron a memory of its previous states and thus prevents the neuron from changing its state instantly. The parallel resistor implements a "forgetting" term, such that if all the inputs to a particular neuron are zero then the activity will decay.

The equation below describes the dynamic behaviour of such a neuron (Figure 2.7).

$$\frac{dx_i}{dt} = -\frac{x_i}{\tau} + \sum_{j=1}^{N} V_{Tij} V_j + I_i \qquad (2.8)$$

where

x_i - the neural activity of neuron i

V_i - the neural state of neuron i

I_i - the bias current for neuron i

$V_{Tij} = \frac{1}{R_{ij}}$ - the synapse weight from neuron j to neuron i

$\frac{1}{R_i} = \frac{1}{\rho_i} + \sum_{j=1}^{N} \frac{1}{R_{ij}}$

$\tau = R_i C$ - system time constant

The sigmoidal relationship between the activity voltage, x_i, and the neuron output state, V_i, is defined as

$$V_i = \frac{1}{2}(1 + \tanh(\frac{x_i}{x_0})) \tag{2.9}$$

The weights and bias currents specify the energy function for the network.

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} V_{Tij} V_i V_j - \sum_{i=1}^{N} V_i I_i \tag{2.10}$$

Dynamically, the feedback in the network acts to minimise the total energy of the network. Thus by encoding an optimisation problem in terms of this energy (cost) function the Hopfield/Tank will produce a "low" cost solution to the problem. Unfortunately, because the dynamic motion of the network is equivalent to gradient descent the system can converge to solutions which are local rather than global minima.

The example Hopfield/Tank chose to demonstrate the power of their network was the 10-city TSP. They represented the problem using a 10×10 grid of neurons with one axis representing the cities and the other representing the position of a city in the tour (Figure 2.8). In general, the N city problem maps on to N^2 neurons and N^4 synapses. The energy equation constructed favoured short tours and contained terms which ensured that tours satisfying the constraints of visiting all cities once and once only corresponded to the low energy states of the network (see Section 2.5.2 for a more detailed description).

Their results claim a 80% success rate (i.e. 80% of all runs they carried out converged to a valid tour). Statistically, this may not be highly accurate as the experiment was only performed 20 times.

Several attempts to reproduce these results have met with widely varying degrees of success. Wilson and Pawley [15] found only a 8% success rate, but found later that they had set the system time constant to 1 instead of 10^{-4}. Paielli [16] found a 78% success rate while Kamgar-Parsi *et al* [17] initially only found a 4% success rate. However this

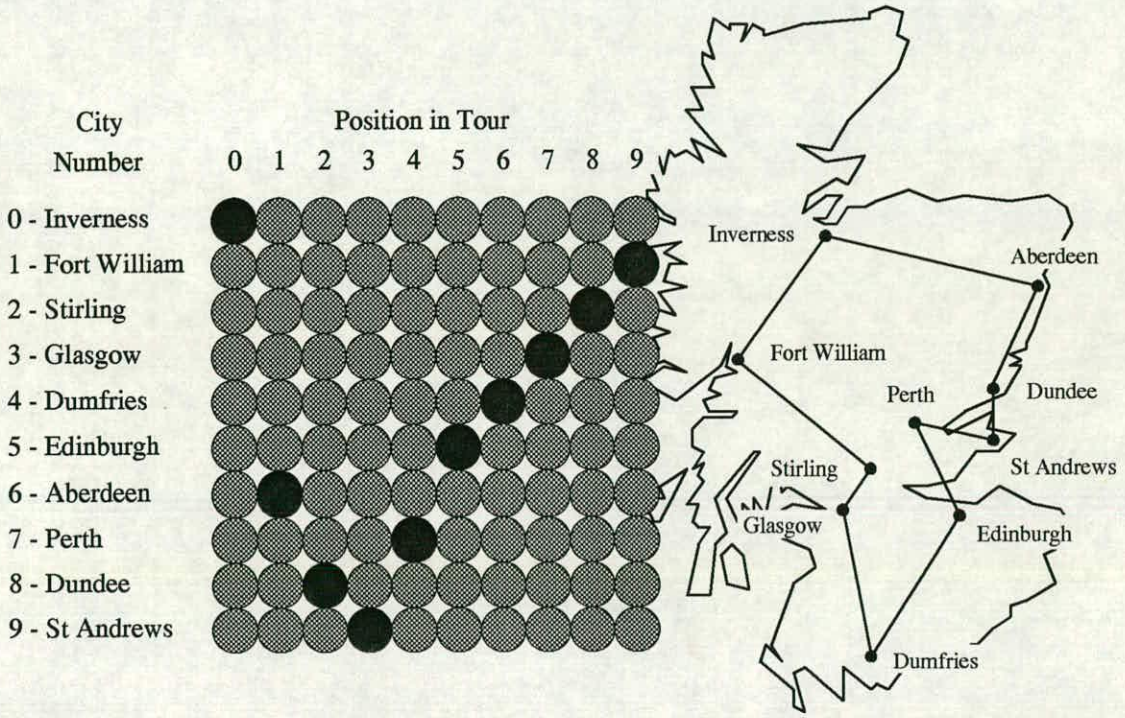


Figure 2.8 The Network Structure for the TSP.

was improved to 33% and finally 50% after a series of modifications to the energy function.

The conclusion must be that the Hopfield/Tank network is extremely sensitive to the implementation, the values of the coefficients of the terms in the energy function, and perhaps even to the coordinates of the cities. This sensitivity to the value of these parameters is further supported by Hopfield and Tank's difficulty in finding a good parameter set for the 30-city TSP. Scaling is also a problem, as the number of synapses increases by N^4 as the number of cities N increases. As a result, large scale problems require very large networks. Also, as the network is scaled up, its sensitivity to its parameters is further heightened. However, valid tours, when they are obtained, are generally of good quality.

In subsequent papers [18, 19] Hopfield and Tank extended the basic structure of their network to the problems of analogue to digital conversion, a linear programming problem and a task assignment problem.

2.4.2. Stochastic Neural Networks

Stochastic neural networks [20, 21] combine the basic structure of a Hopfield/Tank network with a cooling schedule similar to simulated annealing to create a network which avoids becoming trapped in local minima. The neuron is the same as a Hopfield/Tank neuron except that it also contains a controllable noise source. The magnitude (temperature) of this noise source is initially large, and as time elapses the system is cooled slowly to allow the network to converge to a valid solution. This prevents the network from becoming trapped in local minima by allowing the network to jump over energy barriers, thus improving the quality of the solution obtained. The system must be cooled very slowly to guarantee that the optimal solution is found. As a result long compute times are required. If the system is cooled more rapidly to reduce the computer run time required, the network is more likely to become trapped in a local minimum.

2.4.3. Boltzmann Machines

The fully interconnected structure of a Boltzmann Machine [22, 23, 21] is similar to the topology of the Hopfield/Tank neural network with symmetric weights ($V_{Tij} = V_{Tji}$). The main differences are the use of neurons with binary outputs and a probabilistic update rule. To update a unit, the energy difference in the system with the unit turned on and off is calculated. Owing to the symmetric connections, this energy difference is given by the sum of the synaptic multiplications minus the threshold value, θ_i, of that unit (see Equation 2.11).

$$\Delta E_i = \sum_{j=1}^{N} V_{Tij} V_j - \theta_i \tag{2.11}$$

Equation 2.12 then uses this change in energy to determine the probability that the unit is turned on.

$$P_i = \frac{1}{1 + e^{\frac{-\Delta E_i}{T}}} \tag{2.12}$$

where T = Temperature of the network.

If this probability is greater than a random number from a uniform [0.0,1.0] distribution then the unit is set to a 1, otherwise it is set to 0. By repeating this step, the energy of the system is minimised. The calculation of the energy difference in the system between a unit turned on and off is only valid if one unit is updated at a time.

To prevent the network from becoming trapped in local minima a Simulated Annealing cooling schedule is used. At the outset the temperature, T, is high allowing the network to jump out of local minima. T is subsequently reduced slowly such that the network converges, it is hoped, to the optimal solution.

Despite the titles containing the phrase "Stochastic Neural Network for Solving Job-Shop Scheduling", the papers by Foo and Takefuji [24, 25] describe a Boltzmann Machine scheduling system. The Boltzmann Machine implementation is slightly different to that described earlier as there is an additional level of feedback which monitors the variation of the cost function. If the function increases or remains constant then the feedback increases the probability that the units will change state to try to move into a more favourable solution space. For the given example the system produced 10 different schedules including the optimal one.

Gutzmann [26] and Cervantes [21] have also used a Boltzmann Machine to solve successfully combinatorial optimisation problems.

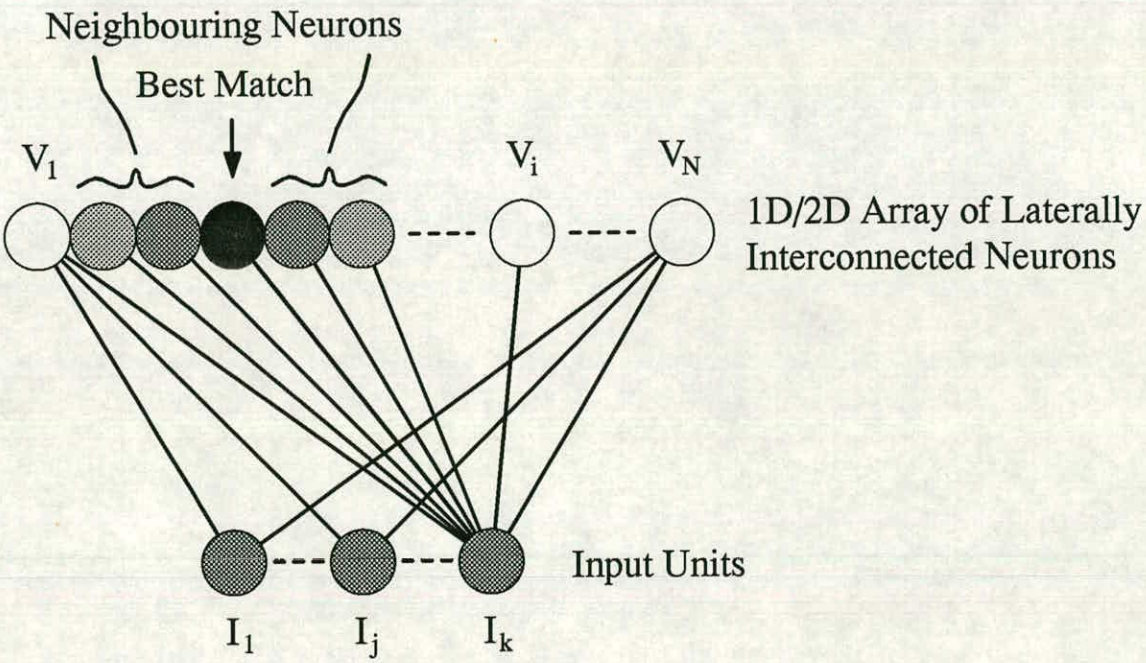


Figure 2.9 The Kohonen Self-Organising Neural Network.

2.4.4. The Kohonen Self-organising Neural Network

Until recently the Kohonen self-organising network has been applied largely to clustering problems, to infer the internal relationships between the vectors making up a block of data. In the last few years several papers [27, 28, 2, 29] have reported the use of the Kohonen neural network to solve the optimisation problems of cell placement in VLSI circuits, scheduling and the Travelling Salesman Problem (TSP).

A Kohonen network [30, 31] consists of a layer of laterally interacting adaptive output neurons and a layer of input cells (Figure 2.9). Each of the input cells is linked to

every output neuron via synaptic interconnections. The structure of the output layer is determined by the problem, but generally most problems are clustered using either a line or an array structure.

The values of the synaptic connections from the input units to one output neuron form a weight vector, w_r. Each output neuron compares the input vector with its weight vector by calculating either the Euclidean or Manhattan distance between the two vectors. The neuron with the smallest output resulting from this parallel correlation is the best match for that particular input vector. Thus a particular input vector is associated with a single output neuron.

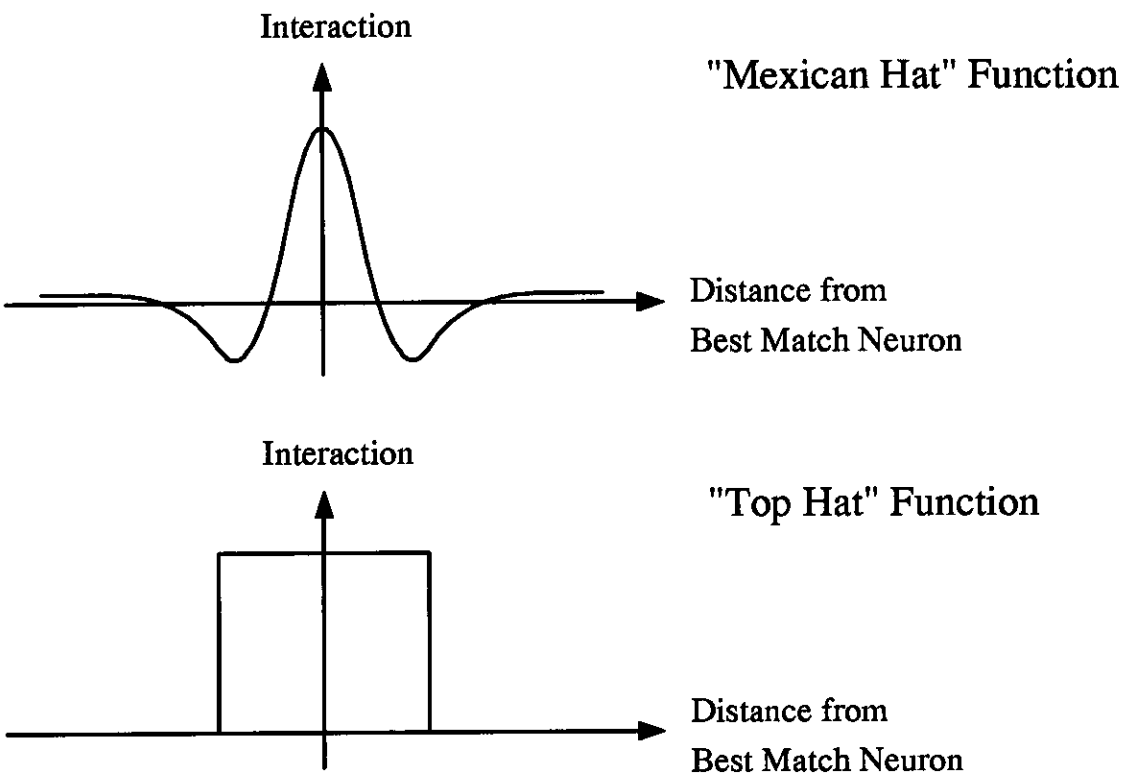


Figure 2.10 The "Mexican Hat" and "Top Hat" Weight Vector Update Functions.

During the self-organising phase, each time a training vector is presented the best match neuron is identified. Its weight vector is then modified to enhance the match between the 2 vectors. The weight vectors of the neighbouring neurons are also pulled towards the input vector. This action forces physically adjacent neurons in the output layer to respond to similar vectors. The degree to which neighbouring weight vectors are updated is determined by either a "Mexican hat" or a "top hat" function centred on the best match neuron (Figure 2.10). By varying the width of this function the size of

neighbourhood updated around the best match neuron can be changed.

As training progresses, the neurons excited by the training vectors become distributed over the array such that similar vectors excite neurons close together and dissimilar vectors excite neurons at the opposite ends of the array. Kohonen demonstrated this effect by training a square array on 2 random number inputs [32]. At the end of this training phase the array had organised itself to respond like a set of x-y coordinates. The neuron at the left-hand lower corner of the array was excited by a (0,0) input while (1,1) excited the neuron at the top right-hand corner.

Angeniol *et al* [2] and Yoshihara *et al* [29] use a Kohonen network with its output neurons arranged in a ring structure to solve the TSP. The starting point of the tour is therefore related to the last city in the tour by the structure of the network. The number of neurons in this ring is typically 2 to 4 times the number of cities for which a solution is sought, depending on the particular city data set. If the Euclidean update rule is used, 2 input units are required while the alternative scalar (dot) product implementation needs 3 input units. The training vectors for the network are simply the coordinates for the N cities. Graphically the ring of points defined by the weight vectors is expanding out under the self-organisation process to fit the positions of the cities. Minimisation of the tour length is achieved by the clustering action of the organising algorithm, which tries to place cities which are close together on adjacent neurons in the output. This algorithm is very similar, in concept, to the elastic net method devised by Durbin and Willshaw [33] for the TSP. In the elastic net method a set of points describing a closed path (initially a small circle) expands out to fit the positions of the cities. This expansion is controlled by two forces. The first force pulls a point on the path towards its nearest city. The second force pulls a point towards its neighbours on the path, minimising the overall length of the tour.

This particular neural encoding of the TSP is very compact. The 100 city problem on a Kohonen network using the Euclidean squared update rule only requires 200 neurons and 400 synapses. However, for the same size of problem the Hopfield/Tank network requires 10,000 neurons and 100,000,000 synapses.

Hemani and Postula [28, 27] mapped scheduling and VLSI cell placement problems onto the Kohonen network via a connectivity matrix, which specifies connections between either the cells or the jobs to be scheduled. The columns of the matrix then form the training vectors of the network. Cells sharing common connections have similar column vectors. The clustering action of the Kohonen network groups such cells close together, thus minimising the distance between connected cells.

2.4.5. Conclusions

The Kohonen network appears to solve optimisation problems quickly, and to produce optimal or near-optimal solutions. It also displays none of the parameter sensitivity and scaling problems which afflict Hopfield/Tank style networks.

The use of simulated annealing techniques in the Stochastic and Boltzmann networks should, in theory, guarantee an optimum solution if the system is cooled over an infinite period of time. However the Hopfield/Tank network requires many runs from different start points before a good solution can be guaranteed.

Conversely the Stochastic and Boltzmann networks converge more slowly than the pure Hopfield/Tank network due to the effects of a very slow annealing schedule. Thus there is a trade off between the number of runs required to produce a good solution and the time taken for the network to converge. Boltzmann machines are slower than Stochastic networks as neurons are updated one at a time.

2.5. Simulation Results : Neural Networks Applied to the TSP

The traditional optimisation techniques outlined in Section 2.3 have already been researched extensively and as a result their characteristics are well established. In contrast neural networks are relatively new in the optimisation arena and the algorithms are still being developed. Nevertheless, some very promising results for optimisation neural networks have been reported. They appear to have the potential to find optimal or near optimal solutions to a range of optimisation problems relatively quickly. Thus it was decided to investigate the performance of neural networks with a view to a dedicated VLSI implementation. Without a completely parallel neural implementation, the computation of a network becomes serialised, and the maximum potential speed of a network is not fully exploited.

The decision about which of the four optimisation neural networks outlined in Section 2.4 to implement in silicon was influenced by two factors.

1 For efficient VLSI implementation, neural algorithms should be as simple as is possible.

2 Within this research group, funding limitations allowed three researchers only one large scale VLSI implementation. The other researchers required a VLSI implementation to accelerate the performance of Multilayer Perceptron (MLP) networks.

The first point effectively rules out Boltzmann Machines and Stochastic Neural Networks as it is difficult to implement uncorrelated, controllable noise sources within an integrated circuit. To satisfy the second constraint the architecture of the integrated circuit had to conform closely to the basic neural model outlined at the start of Section 2.4.

Given these two constraints the Hopfield/Tank and the Kohonen neural networks were the most appropriate optimisation neural networks for this particular neural VLSI hardware implementation.

To obtain statistically significant information about the performance of these two networks, a series of software simulations was performed. The simulations also provided a base line for comparing the results obtained from the VLSI networks discussed in Chapter 6. The ubiquitous TSP was chosen as the test problem simply because there was so much information available about how other optimisation techniques performed on this problem.

Processing file:	city10_0.dat
Number of Cities:	10
Time to find max, mean, and min:	75 s
Time to calculate PDF and Standard Deviation:	192 s
Number of Tours:	362880
Minimum tour length:	2.690670
Average tour length:	4.765494
Maximum tour length:	6.288088
Variance:	0.241941
Standard deviation:	0.491875
Optimum Tour is in file:	city10_0.all.optimum
Optimum Tour Path:	0 3 4 5 6 7 8 9 1 2
Probability Distribution Function file:	city10_0.all.pdf
Area under PDF:	362880

Figure 2.11 An Example Output from the Tree Search Program.

2.5.1. An Exhaustive Search of the Solutions to the TSP

To assess the success of the Hopfield/Tank and Kohonen Self-Organising neural networks, the full distribution of possible valid tours for the city set being used must be known. A tree search programme was written in 'C' to find the optimal solution and the distribution of the $(N-1)!/2$ possible solutions for the N city problem.

Hopfield/Tank's choice of the 10 city problem is an interesting one. For larger problems, the computational time required for an exhaustive search becomes prohibitive. For example, as Figure 2.11 shows, the exhaustive search technique takes 75s to find the optimal solution on a Sun IPC Sparc workstation. Using this as a basis gives the times

required for an exhaustive search for the 15 and 50 city problems as 208 days and 10^{51} years respectively. To put the last of these figures into perspective the age of the universe is estimated at about 10^{10} years. This illustrates graphically why exhaustive search is impractical for medium to large problems.

The exhaustive search program uses a recursive subroutine to carry out a tree search of all the possible tours. The program reduces the size of the search tree by making use of the fact that the tour length is independent of the start point. However it does not use the additional information, that the tour length is also independent of direction of travel. This would result in the recursive subroutine becoming overly complex and slow.

The tree search program was used to calculate the optimal tours and the probability density function for the 10 city example Hopfield/Tank used and 100 randomly generated 10 city data sets. Figure 2.11 shows an example output from the tree search program for the Hopfield/Tank city set and Figure 2.12 shows the distribution of all the valid tours for that city set.

Figure 2.11 shows that the optimal tour for the 10 city data set has length 2.691; the optimal tour is depicted in Figure 2.13. This is the same optimal tour that was reported by Hopfield/Tank.

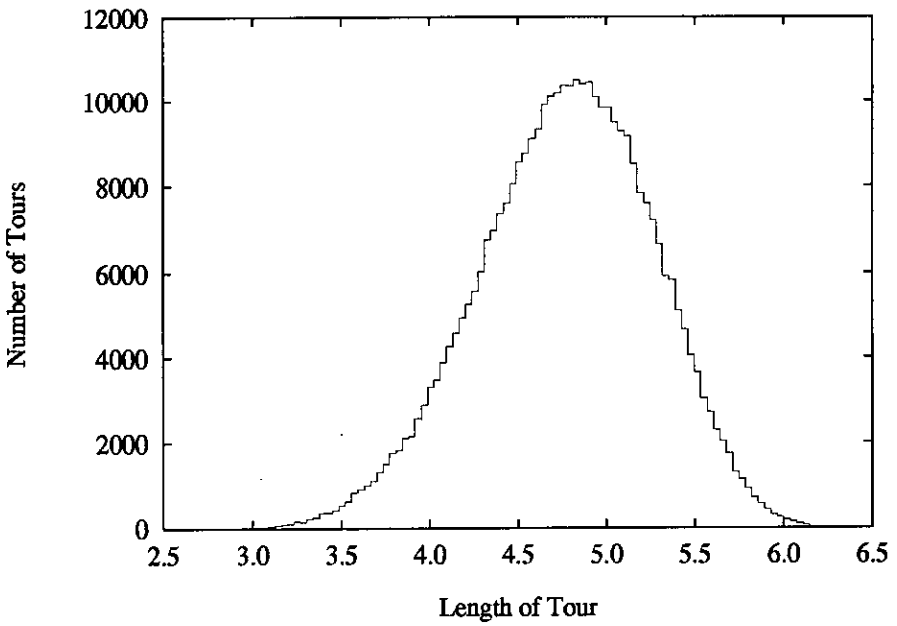


Figure 2.12 The PDF for the Hopfield/Tank 10 City Example.

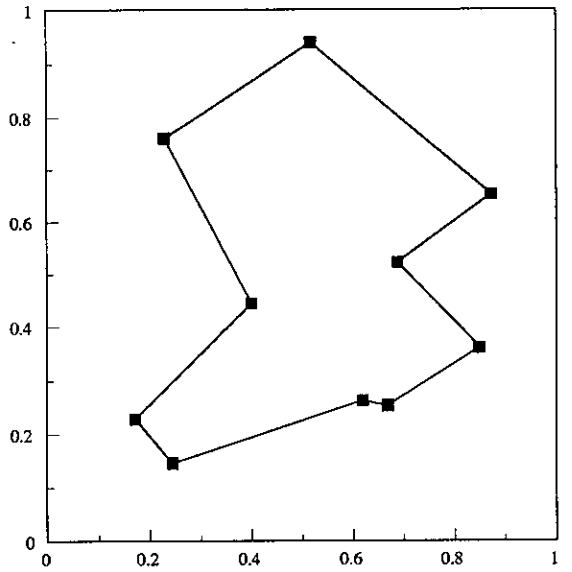


Figure 2.13 The Optimum Tour for the Hopfield/Tank 10 City Example.

2.5.2. The Hopfield/Tank Neural Network Applied to the TSP

As is shown in Figure 2.8 the Hopfield/Tank neural network uses a N by N array of neurons to solve the N city TSP. One axis represents the cities and the second the positions of those cities in the tour. Thus the state value of a neuron, V_{Xi}, represents whether city X is occupying position i in the tour. The energy function for the TSP (Equation 2.13) uses the structure of the square array to encode the constraints such that only one city can be visited at a time and that each city should be visited once and once only. This translates into the restriction that only one neuron in any row/column combination should be on, to ensure that the tour produced is valid. The first two terms in Equation 2.13 implement these two constraints.

$$E = \frac{A}{2}\sum_{X}^{N}\sum_{i}^{N}\sum_{j\neq i}^{N} V_{Xi}V_{Xj} + \frac{B}{2}\sum_{i}^{N}\sum_{X}^{N}\sum_{Y\neq X}^{N} V_{Xi}V_{Yi} + \frac{C}{2}\left(\sum_{X}^{N}\sum_{i}^{N} V_{Xi} - n\right)^2 + \frac{D}{2}\sum_{X}^{N}\sum_{Y\neq X}^{N}\sum_{i}^{N} d_{XY}V_{Xi}(V_{Y,i+1} + V_{Y,i-1}) \quad (2.13)$$

where

d_{XY} - the distance between cities X and Y.

A, B, C, D and n are parameters defining the relative importance attracted to each term.

The third term ensures that only N cities are visited. The last term biases the network to favour short tours by making them the low energy states of the system. This also means that the weight set is dependent on the positions of the cities in the tour.

As the Hopfield/Tank network essentially performs gradient descent on the weight set derived from the energy function, the start point of the network will have a significant influence on which minimum the network reaches. Hopfield and Tank initialised the neurons to random values subject to the condition that

$$\sum_{X=1}^{N}\sum_{i=1}^{N} V_{Xi} = N \tag{2.14}$$

to give the network a unbiased start point. These values for, V_{Xi}, are then used to determine the neural activity values, u_{Xi}. A small random component is added to the neural activity value to break any possible symmetry in the initialisation, allowing the network to converge to a solution rather than staying in a quasi-stable state.

The values of the parameters used by Hopfield/Tank are listed below:

A	=	500	n^+	=	15
B	=	500	I_{Xi}	=	3000
C	=	200	RC Constant	=	0.0001
D	=	500	u_o	=	0.02

Time Step = 0.00001

One implementation issue is the order in which the neural state values are updated during each time step. Three different update orders were tried.

1 Update each neuron one at a time in a random pattern (Random Update Rule).

2 Update each neuron one at a time in a raster (TV Scan) pattern (Raster Update Rule).

3 Calculate all the activity increments in parallel and then update all of the neurons at the same time (Parallel Update Rule).

The random update rule is the most accurate model of the behaviour for the asynchronous, analogue circuit upon which the Hopfield/Tank network is based. However the parallel update rule is a better model of hardware implementations which comprise a synaptic multiplication cycle followed by neural update cycle.

The simulations explored the effects of these three different update rules on the Hopfield/Tank networks performance. Tables 2.1a, 2.1b and 2.1c show the results of these simulations for the Hopfield/Tank 10 city data set for various values of the simulation time step.

The results indicate that, as the value of the time step is reduced, the performances of the 3 different update rules converge. It is interesting to note that a time step of 0.00001 for the parallel update version is large enough for the network to start oscillating before converging to a solution. This oscillation could be beneficial in allowing the network to jump out of local minima and hopefully into the global minimum. However, the

Update Rule	No of Runs	Valid Tours	Optimal Tours	Min	Mean	Max
Random	2500	1163	137	2.691	2.944	3.842
Raster	2500	1075	214	2.691	2.895	3.728
Parallel	2500	1218	245	2.691	2.900	3.686

Table 2.1a Results for the Hopfield/Tank Network : Time Step = 0.00001.

Update Rule	No of Runs	Valid Tours	Optimal Tours	Min	Mean	Max
Random	2500	1181	165	2.671	2.925	3.738
Raster	2500	1125	188	2.691	2.923	3.742
Parallel	2500	1168	157	2.691	2.936	3.822

Table 2.1b Results for the Hopfield/Tank Network : Time Step = 0.000005.

Update Rule	No of Runs	Valid Tours	Optimal Tours	Min	Mean	Max
Random	2500	1144	163	2.691	2.923	4.055
Raster	2500	1160	146	2.691	2.932	3.949
Parallel	2500	1170	163	2.691	2.935	3.822

Table 2.1c Results for the Hopfield/Tank Network : Time Step = 0.000001.

network is now behaving in a chaotic manner rather than performing gradient descent. The main problem with a small time step is that it increases the number of iterations and thus the time required for the network to converge to a solution. A time step of 0.000005 was chosen as the best compromise between time for the network to converge and the stability of that convergence.

For the parallel update version with the smallest time step, 46.8% of the 2500 runs converged to valid tours and 6.5% of those runs produced the optimal tour. Thus on average the network should produce the optimal tour every 16 runs. The exponential distribution of the probability density function for the valid tours produced by the parallel update (Figure 2.14) confirms that the majority of the tours produced are indeed good tours.

It is not possible to quote exact times for computation as the run time to produce a solution is dependent on many variables, viz:

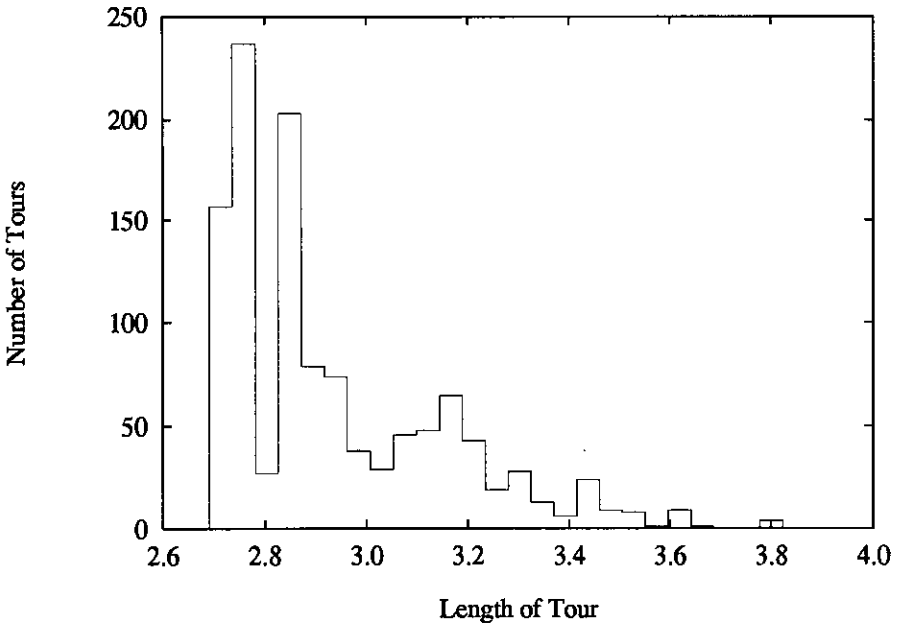


Figure 2.14 The PDF of the Tours produced by the Hopfield/Tank network.

1 The size of the time step.

2 The number of iterations required to converge to a solution varies with the random initialisation.

3 The number of runs required to produce a good solution, which can vary between 20 and 50, depends on the data set.

Table 2.2 contains the average simulation times for one run of the network. The timings for the random, raster and parallel update orders are similar.

Time Step	Average Time for a Run
0.00001	12s
0.000005	24s
0.000001	120s

Table 2.2 Order of Magnitude Timings for the Hopfield/Tank Network

To test the network thoroughly it was applied to the 100 randomly generated 10 city data sets. For each city, its x and y coordinates were randomly generated within a square of unit side. A time step of 0.000005 and the parallel update rule were used as this was

the most realistic comparison with the hardware implementation. The simulation was run 100 times from different random initialisations for each city set to increase the probability of the network finding the optimal tour. The full results are contained in Appendix 1. A summary of the results is as follows:

1 The optimal tour was found in 65 of the data sets.

2 The network failed to find any tour for 3 data sets.

3 The best tour produced for the remaining 32 data sets was always better than the mean tour length of all possible tours.

This confirms that the Hopfield/Tank network does indeed produce good solutions to the 10 city TSP. The failure in 3 cases to find any tour illustrates the Hopfield/Tank's neural network sensitivity to the shape of the energy function. Increasing the size of network increases this sensitivity. Hence, for large scale problems it is very difficult to set the network up correctly.

2.5.3. The Kohonen Self-Organising Network Applied to the TSP

The Kohonen network used to solve the TSP has its output neurons arranged in a ring structure. The starting point of the tour is therefore related to the last city in the tour by the structure of the network. There are two different ways of finding the best match neuron for a particular input vector.

1 The Euclidean Update Rule :

$$d_{i,r} = \sqrt{(x_{i(1)} - w_{r(1)})^2 + \cdots + (x_{i(n)} - w_{r(n)})^2} \tag{2.15}$$

The above rule measures the distance between the points defined by the input vector and a weight vector; the smaller the distance the closer the match.

2 The Dot Product Update Rule :

$$x_i . w_r = |x_i||w_r| \cos\theta_{i,r} = x_{i(1)}w_{r(1)} + \cdots + x_{i(n)}w_{r(n)} = \sum_{j=1}^{n} x_{i(j)}w_{r(j)} \tag{2.16}$$

The dot product update rule determines the proximity of two points by calculating the cosine of the angle between the vectors which define the two points, $\cos\theta_{i,r}$.

If both vectors have unit magnitude then the value of the dot product is directly proportional to the value of $\cos\theta_{i,r}$; the larger the value of the cosine the better the match.

To translate the 2 dimensional city coordinates into vectors of unit magnitude, the 2 dimensional square containing the city coordinates is mapped onto the surface of a sphere with unit radius. The mapping is achieved by first

compressing the unit square containing the city coordinates by 0.707 on each side and then projecting each point vertically up onto the surface of the unit sphere (Figure 2.15). There are now 3 inputs to the network, two of which represent the city position and the third which maps the city onto the surface of the sphere. Since the proposed neural hardware consists of an array of multipliers, then the dot product rule is clearly the preferred choice.

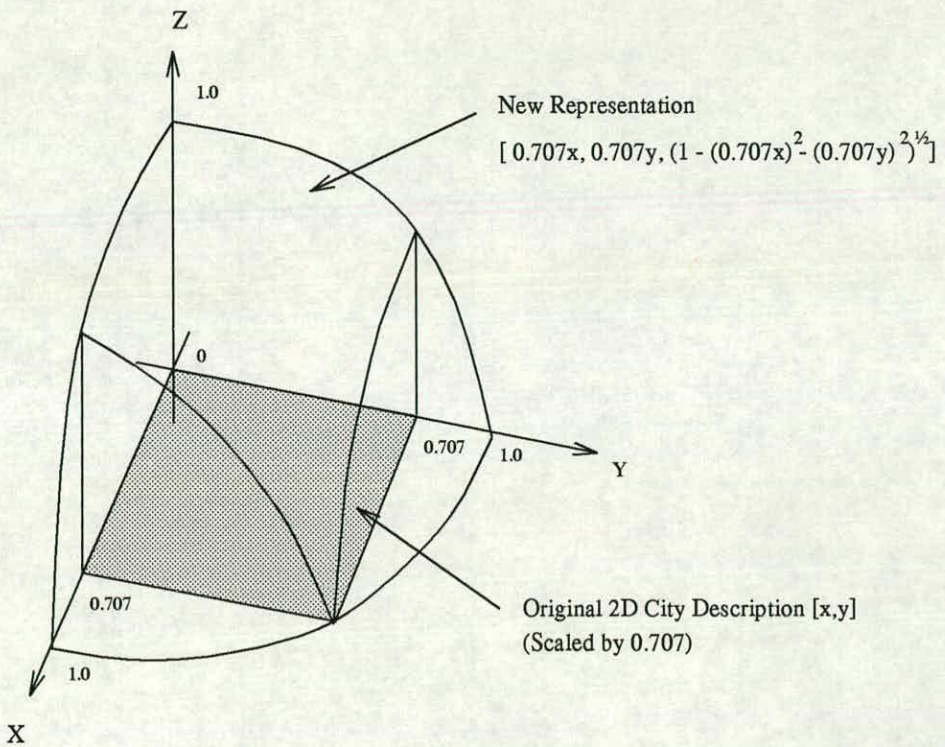


Figure 2.15 The input mapping required for the Dot Product update rule.

The training vectors for the network are simply the coordinates for the N cities. After each presentation of a training vector, the Gaussian neighbourhood function, h_{rs}, (Equation 2.17) centred on the best match neuron determines to what degree the weights of the surrounding neurons are affected.

$$h_{rs} = \exp\left(-\frac{|r-s|^2}{\sigma^2}\right) \tag{2.17}$$

where $|r-s|$ is the distance between the best match neuron, s, and neighbouring neuron, r, along the circumference of the ring of output neurons. The variance σ^2 controls how far the arc of the neighbourhood extends along the ring of output neurons.

The weight vectors are then updated according to Equation 2.18.

$$w_{r(new)} = w_{r(old)} + \varepsilon . h_{rs} . (x - w_{r(old)}) \tag{2.18}$$

where ε is the learning rate of the network.

After every complete presentation of the set of training vectors (epoch) the size of the neighbourhood is reduced, until eventually the weights of only one neuron, the closest match neuron, are modified. At this point the self-organisation process is complete.

Equation 2.19 is used to calculate the correct value for σ based on the desired initial and final neighbourhood sizes and the number of epochs elapsed.

$$\sigma(t) = \sigma_i \left(\frac{\sigma_f}{\sigma_i} \right)^{\frac{t}{t_{max}}} \tag{2.19}$$

where

$\sigma(t)$	=	neighbourhood size at epoch t.	t	=	number of elapsed epochs.
σ_i	=	initial neighbourhood size.	t_{max}	=	maximum number of epochs.
σ_f	=	final neighbourhood size.			

After training is complete, the neuron which is the best match when a city is presented to the network represents its position in the tour. The network orders the cities in such away that the distance between them is minimised. Apart from the structure of the output nodes, the rest of the operation of the network is identical to a Kohonen network used for clustering [30].

Initial simulations of the Kohonen Self-Organising Neural Network suggested that the number of neurons required was 2 to 3 times the number of cities, to allow the network to assign one individual neuron to every city. If the number of neurons used is too small then the clustering action of the Kohonen network will tend to group cities which are close together under one neuron. The rest of parameters for the network are given below.

σ_i	=	N	ε	=	0.3
σ_f	=	0.5	t_{max}	=	100

where N is the number of cities.

To find if there was any difference in the quality of tours produced by the two different update rules, a comparison was run on 10 randomly generated 50 city data sets. The results from this trial (Table 2.3) show that while the tours they find are indeed different, there is not a systematic difference in terms of tour quality. The mapping from a 2 dimensional unit square onto a the surface of a unit radius sphere maps the (0,0),(1,1) main diagonal of the unit square on to an arc of length $\pi/4$ and the (0,1),(1,0) diagonal on to arc of length $\pi/6$. Thus the mapping is distorting the original x-y relationship between the cities by stretching the surface defined by the unit square along the (0,0),(1,1) diagonal.

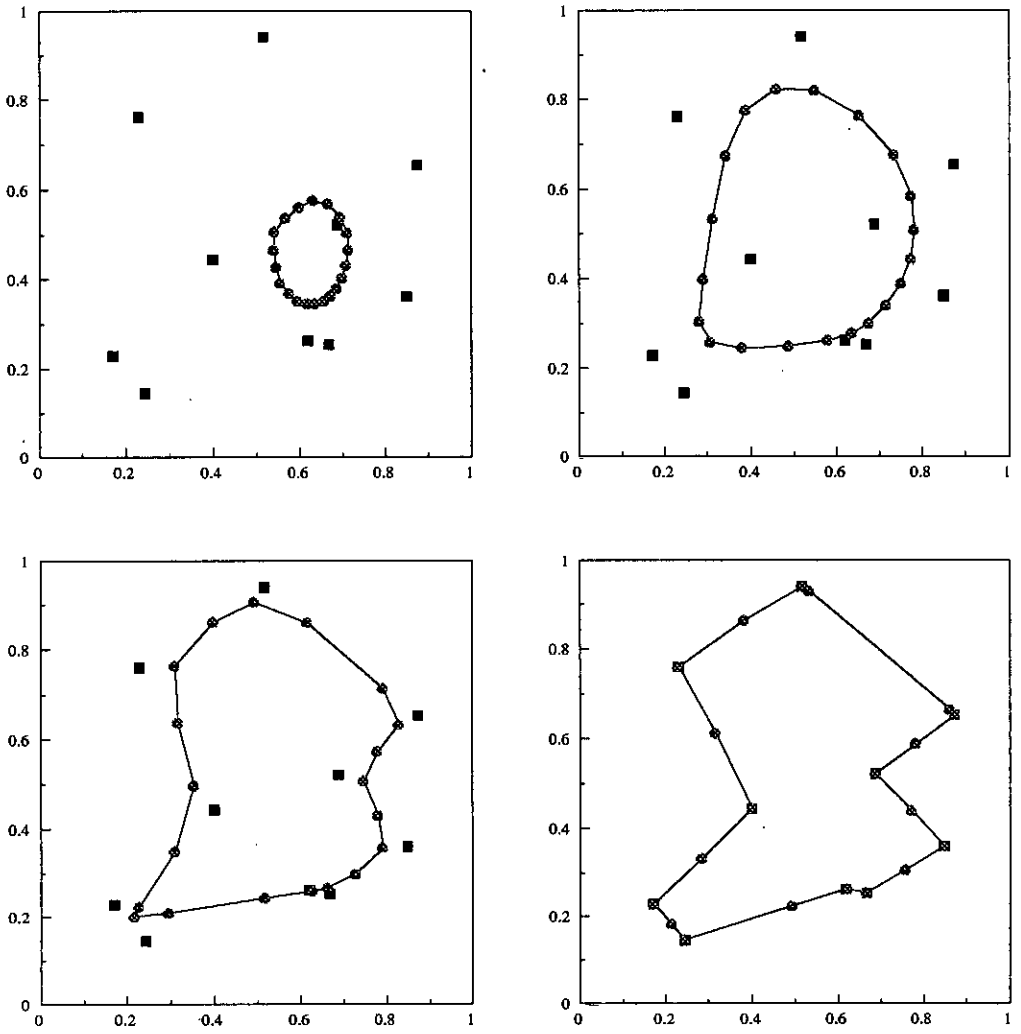


Figure 2.16 The Solution of the 10 City TSP Using a Kohonen Network.

As the results shown in Table 2.4 demonstrate, this network produced the optimal solution to the Hopfield/Tank 10 city example city set in 2s. Figure 2.16 shows the tour path for this example as it develops with the number of epochs. This clearly demonstrates the similarity between the Kohonen network and the Durbin and Willshaw elastic net method. Further simulations confirmed that this particular implementation of the Kohonen network with the initial neighbourhood size covering all of the ring of neurons, was immune to the initial values of the weights. The large initial neighbourhood size has a global averaging effect which always moves the ring of neurons to approximately the same initial position after the first few epochs.

City Set	No of Cities	No of Neurons	Tour Length	
			Euclidean	Dot Product
rand0_50	50	100	6.347	6.657
rand1_50	50	100	6.134	6.181
rand2_50	50	100	6.022	5.992
rand3_50	50	100	6.036	6.347
rand4_50	50	100	5.724	5.893
rand5_50	50	100	6.153	6.116
rand6_50	50	100	6.133	5.969
rand7_50	50	100	5.529	5.516
rand8_50	50	100	5.992	6.101
rand9_50	50	100	6.043	6.038

Table 2.3 Comparison of Euclidean and Dot Product update rules.

To find out how often the network found the optimal tour, it was applied to the 100 randomly generated 10 city data sets. The results are shown in Appendix 1 alongside the results for the exhaustive search and Hopfield/Tank optimisation techniques. Out of the 100 city sets, the network found the optimal tour for 73 of the city sets and in the other 27 the tour produced was close to the optimal one. Thus it can be seen that the network typically produces very good tours with the bonus that the optimal tour is often found. It should also be noted that this network cannot produce invalid tours due to the structure of the network.

City Set	No of Cities	No of Neurons	Tour Length	Time (s)
city10_0	10	20	2.691	2
rand10_00	10	30	2.450	3
rand50_0	50	100	6.657	47
rand100_0	100	300	8.714	274
rand1000_0	1000	3000	25.573	31485

Table 2.4 Timings for the Kohonen Solving the TSP.

The next stage was to explore how the performance of the Kohonen network scaled with the number of cities. To this end the network was applied to 50 city, 100 city and

1000 city randomly generated city sets. Table 2.4 shows the time the network took to solve each problem size. The computation time is proportional to the number of cities, N, multiplied by the number of neurons, 2N to 3N.

2.5.4. Conclusions

From the above simulations it became clear that there are 3 main criteria for accessing the performance of a network.

1 Quality of the solutions produced.

2 Speed of computation.

3 Scalability.

Both networks score well according to the first criterion.

1 The Hopfield/Tank network in general produces good tours. If the network is run many times, then there is a good probability that the best tour produced will indeed be the optimal one.

2 The Kohonen network always converges to a good tour and for 73% of the 100 randomly generated city sets it produced the optimal tour.

It is when applying the second and third criteria that the differences begin to appear.

1 The Kohonen takes only 3 seconds on a Sun IPC Sparc workstation to solve the 10 city TSP.

2 The Hopfield/Tank network takes between 10 and 340 seconds for just one run, depending on the time step used and on how quickly the run converges to a solution. The Hopfield/Tank network typically requires to be run 20 to 50 times to guarantee finding a good tour, giving a total run time of typically 1 hour and 40 minutes.

Thus the Kohonen network is 3 orders of magnitude faster. Even if a larger time step was used Kohonen would still be 2 orders of magnitude faster.

The final criterion is how well the network scales with the size of the problem. The main problem with the exhaustive search technique is the computation time required. Table 2.5 shows how the 3 techniques outlined compare in speed and problem size.

Thus the Kohonen network with its small computational load and the smallest rate of increase is the clear winner in speed and scale terms. While estimates have been made for the 50 city, 100 city and 1000 city TSP for the Hopfield/Tank network it is debatable whether networks of this size would be stable due to the sensitivity of the Hopfield/Tank network to the values of the parameters of the energy equation.

However, when these networks are applied to other types of optimisation problems such as task-assignment and scheduling, the advantages of the Kohonen network become

Network	Computational Load	Number of Cities			
		10	50	100	1000
Exhaustive Search	$(N-1)!$	75s	10^{51}years	10^{145}years	10^{2000}years
Hopfield/Tank	N^4	120s	20.8 hours	13.9 days	380 years
Kohonen	N^2	3s	47s	268s	27321s

Table 2.5 A Comparison of Computational Loads.
N.B. The timings for the 50 city, 100 city and 1000 city TSP for the Exhaustive Search and Hopfield/Tank network are estimates based on the timings for the 10 city problem. Stirling's Approximation to the Gamma Function was used to estimate the factorial of 1000 (Appendix 2).

less clear. The difficulty lies in how to map other optimisation problems on to the network. With Hopfield/Tank the rows and columns can be used to represent the available machines/resources versus either the jobs to be carried out or time slots. In the Kohonen network the whole action of the network is to group similar vectors together in the output layer. So, for example, scheduling information has to be somehow transformed into a set of training vectors in which the vectors which need to be close together are separated by small Euclidean distances. For the VLSI cell placement problem, Hemani and Postula [27] used a connection matrix to represent the links between cells, the columns from which were used as the training vectors. While this approach worked for this particular problem it is still difficult to see how to encode the information such that if event i occurs before event j then there is an associated cost c_{ij}, into the training vectors.

In the case of the TSP, the Kohonen Self-Organising neural network has the best performance, with its combination of a small computation load, good characteristics when scaled and the ability to produce very high quality tours. It should be noted, however, that the Kohonen network and the TSP are extremely well suited to each other. The inferior performance of the Hopfield/Tank network is due to the higher computational load of its N^2 neurons and N^4 synapses, and sensitivity of the network to both the value of the input data and the coefficients of the energy function.

Chapter 3

VLSI Implementations of Neural Networks

Due to the serial nature of Von Neumann style computers (for example a Sun IPC Sparc workstation or an IBM Personal Computer (PC)) the parallel computations of a neural network inevitably become serialised. Thus the speed of the massively parallel computation which characterises a neural network is not exploited. This applies equally to all neural networks and not just to the Hopfield/Tank and Kohonen neural networks discussed in the last chapter.

Another factor which slows down the speed of software simulations is that the processor inside either a Sun Sparc workstation or an IBM PC, is designed as a flexible computing engine rather than a high performance multiplier. This means that a multiply-accumulate can take up to 10's of clock cycles. As a result, the use of a specialist processor such a Digital Signal Processor (DSP) which is optimised for performing the multiply-accumulate operation, will yield a considerable gain in performance. The limitations of conventional computers mean that software simulations of neural networks are often unacceptably slow.

All of this points toward the need for a custom VLSI neural network implementation based on a parallel architecture built around dedicated multipliers. Without such a system the true computational power of a neural network cannot be realised.

A custom implementation also allows the architecture and the operations it performs to be optimised for particular neural algorithms which only require, for example, 1-bit neural states (Boltzmann Machines) instead, of 16- or 32-bit floating point numbers. Custom silicon also offers the opportunity to build complete representations of simple biological networks and networks which process data without the supervision of a host computer. The ultimate aspiration is that of a fully parallel, autonomous neural system.

This chapter reviews the main techniques which have been used to implement neural networks in silicon. The digital and analogue VLSI hardware described illustrates the wide diversity of neural VLSI implementations.

3.1. The Great Debate: Digital Versus Analogue

A brief look at some of the reviews [29, 34-37] of VLSI implementations of neural networks highlights two generic implementation styles: digital and analogue. The debate about which technique is better suited to neural network implementation is far from

resolved and the answer varies from research group to research group. However, the decision is largely determined by which technique is the most appropriate for the end application.

Digital multipliers give very fast multiplication times, and the result of the multiplication is totally "process-independent". This reliability together with arbitrary multiplication accuracy and flexibility make digital circuits attractive to a VLSI designer. However, digital multipliers, either parallel or bit-serial, require a large silicon area. Thus it is only possible at present to have tens of digital multipliers on a single integrated circuit.

The compact nature of an analogue multiplier allows thousands of multipliers to be implemented on a single chip. As a result, analogue multipliers have a superior multiplication speed per square millimetre of silicon (speed/area product). However, analogue circuits are much more sensitive to the problems of process variations and noise than their digital counterparts. The massively parallel nature of the structures that form neural networks serves to compound the process variation problem as several chips may be required to implement a particular network.

The debate is further complicated by the existence of hybrid implementations which attempt to combine the best features of digital and analogue circuits. Examples of these include Multiplying Digital to Analogue Converters (MDAC's), switched-capacitor circuits and pulse-based implementations. In general the attempts to combine the best features of each technique are not completely successful; for example in pulse-based implementations, the robustness of digital signal levels is combined with the compactness of an analogue multiplier but the speed of computation is slower than either conventional analogue or digital multipliers.

Ultimately the decision on which is the best implementation style depends on the end application, the type of network used and the interface to the host system. If a general-purpose neural accelerator card for a conventional computer is envisaged then a digital implementation may be the better choice. However, an analogue VLSI implementation would be more suitable for a remote system monitoring the outputs of analogue sensors such as strain gauges.

Sections 2.3 through to 2.6 discuss digital, analogue, on-chip learning and pulse-based VLSI implementations respectively.

3.2. Digital VLSI Implementations

To multiply 2 N-bit numbers, a fully parallel digital multiplier requires N^2 full 1-bit adders (Figure 3.1) while a bit-serial digital multiplier needs only one adder cell (Figure 3.2). For these two N-bit numbers the bit-serial multiplication will take 2N clock cycles. However, the speed of a parallel multiplier is only limited by the time the signals take to

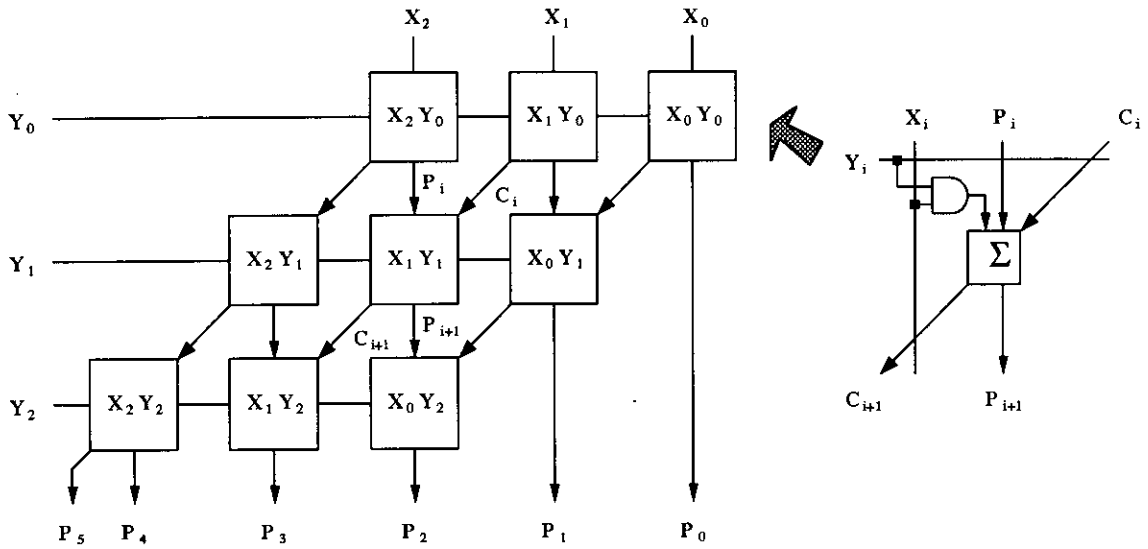


Figure 3.1 Structure of a Fully Parallel Digital Multiplier.

propagate through the logic. Thus there is a direct trade-off between silicon area and speed of operation.

As digital implementations, in general, only contain enough processors to compute a part of the synaptic array during each cycle, the supporting hardware is very important to ensure that the weight and state data are piped as quickly as possible through the multipliers. Section 3.2.4 discusses two general strategies for supplying data to the multipliers: broadcast and systolic.

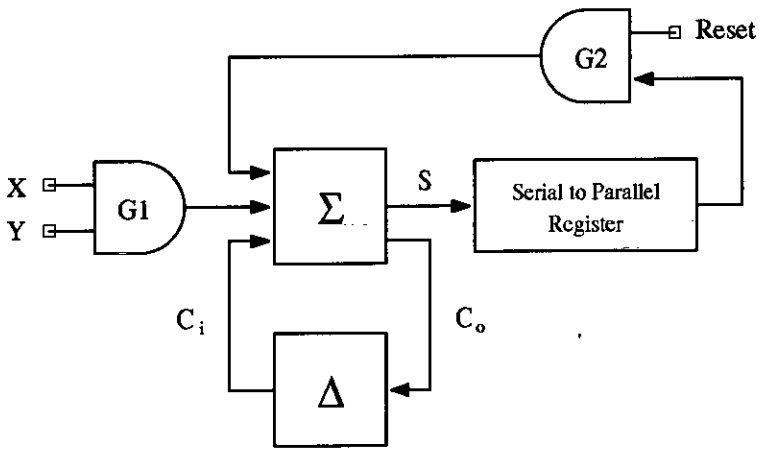


Figure 3.2 Schematic of a Bit-Serial Multiplier.

3.2.1. Digital Signal Processors

Although Digital Signal Processors (DSP's) were originally designed to speed-up the implementation of correlators and discrete Fourier transforms, the optimisation of the multiply accumulate operation in these processors also makes them very suitable for speeding up neural network simulations. Parallel digital multipliers and extensive pipelining of data allow the average DSP (Texas Instruments TMS 32020 or Motorola DSP 56000) to perform a 16-bit by 16-bit integer multiplication plus the accumulate operation in 100ns. Higher specification DSP's like the Motorola 96000 series [38], are able to carry out a 32-bit floating-point multiplication and accumulate operation in only 50ns. While these processors can perform a multiplication much faster than most conventional microprocessors, a custom VLSI array of digital multipliers will be even faster due to 10's of multiplications being carried out in parallel.

To achieve some of this desired parallelism Means and Lisenbee at HNC Inc have developed the SIMD Neurocomputer Array Processor (SNAP) which contains four 32-bit floating point processors [39]. Each of these cells carries out the 32-bit multiplication and accumulate operation in 50ns. The compute power can be increased up to 1,280 MCPS ‡ by cascading 16 SNAP chips (64 processing elements) in parallel. The resultant system is controlled by an Intel i860 RISC processor and linked to the host computer system via a VME communications bus.

Other approaches include simply cascading a number of "off-the-shelf" DSP chips in parallel. For example the Morgan *et al* Ring Array Processor (RAP) board is made up of 4 Texas Instruments TMS320C30 floating point DSP's connected in a ring structure [40, 41]. Ten of these boards working together (40 processors) yield a speed of 574 MCPS.

3.2.2. Bit-Serial Multipliers

An interesting example of the implementation of bit-serial multipliers was that of Butler at the University of Edinburgh [42-45]. To reduce further the area required for a digital multiplier the values of the neural states were restricted to 5 discrete levels {-1.0,-0.5,0,+0.5,+1.0} (Figure 3.3) with the weight value represented by an 8-bit two's complement number. The synapse now only has to pass the weight value untouched, divide it by 2 or output zero. This technique reduced the synapse to a single 8-bit shift register plus an adder/subtractor cell for the distributed bit-serial activity summation. Four fabricated devices, each containing a 9 by 3 synaptic array, formed the 12 by 9 array of synapses at the heart of a neural accelerator board controlled from a Sun 3/110 via a

‡ MCPS - Million Connections Per Second where a connection is one multiply and one accumulate operation. One MCPS is equal to 2 MFlops (Million Floating-point operations per second).

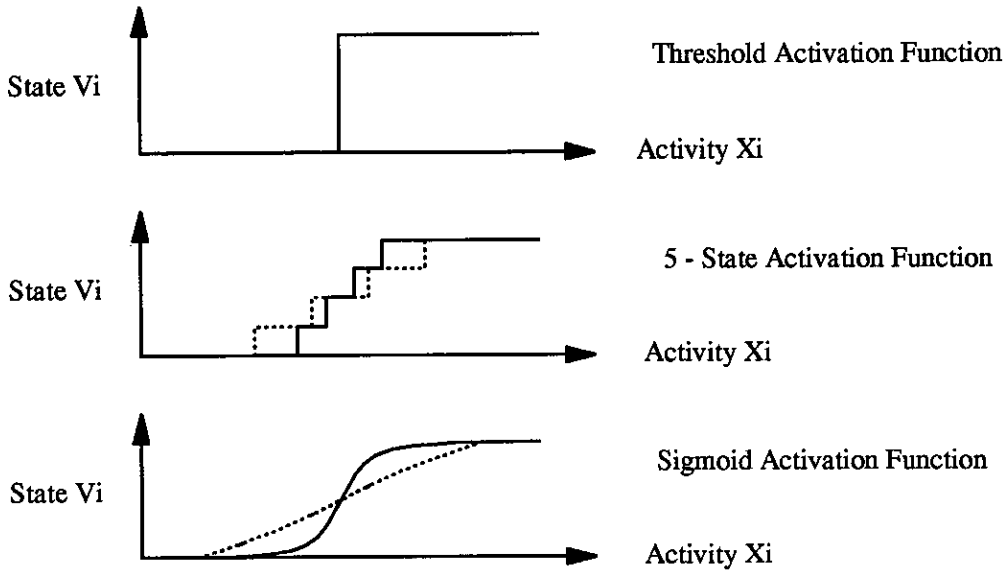


Figure 3.3 Hard Limiter, 5-State and Sigmoid Activation Functions.

VME communication bus. For the pattern classification examples used to demonstrate the system the combination of the Sun 3/110 and the board carried out the multiply-accumulate operation 87 times faster than the Sun 3/110 by itself. While the system was able successfully to perform pattern classification, networks such as MLP's and Hopfield/Tank which require a smooth neural state transition from either -1 to +1 or 0 to +1 are not well suited to this architecture.

A more recent example of a bit-serial implementation is the L-Neuro (Learning Neurochip) chip by Aglan *et al* [46, 47]. To keep the bit-serial multiplier area as small as possible integer arithmetic is used. The configuration of the 1024 eight bit synapses is very flexible. At one extreme it can be set up as a 32 by 16 synaptic array with 16 neurons and 16-bit synaptic coefficients or at the other as a 256 by 4 array with binary inputs. Various intermediate configurations allow neurons to be coded over 1 to 8 bits and synaptic weights as either 8 or 16 bits. Four of the chips, together with a transputer to handle communications, make up a card which slots into the back of an IBM Personal Computer. To increase further the number of processing elements available these boards can also be cascaded. The timings given for recognising handwritten characters using Kohonen Feature Maps show that a single L-neuro device is roughly 8 times faster than a single transputer running at 20MHz.

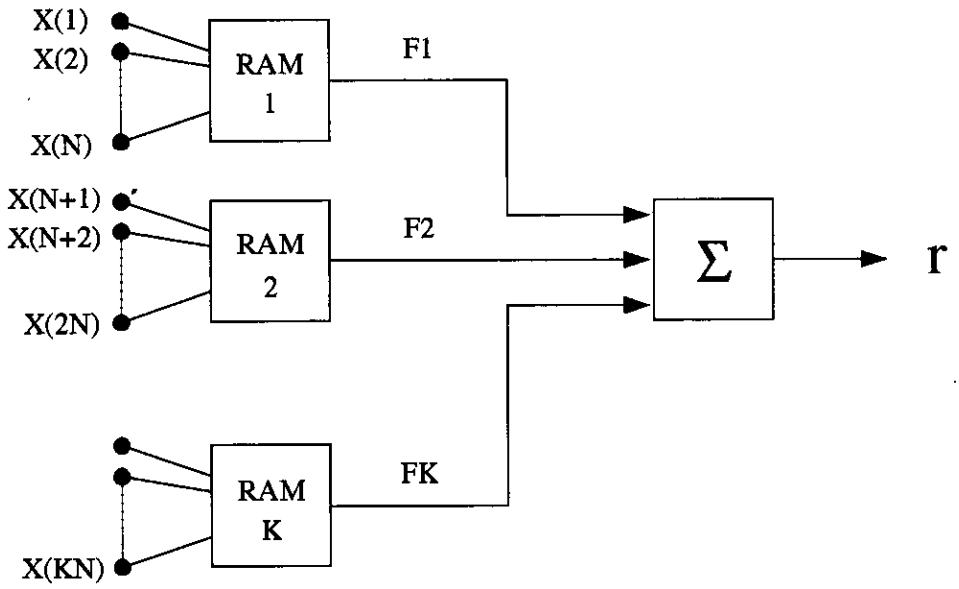


Figure 3.4 WISARD Discriminator.

3.2.3. Specialised Architectures

WISARD by Aleksander *et al* was one of the first neural computing machines to be developed specifically for pattern recognition problems [48, 49]. It differs from other implementations in that it is built of RAM cells rather than arrays of multipliers and summers. For each pattern class there is a discriminator (Figure 3.4). Each of K RAM cells which make up the discriminator looks for a particular feature of that pattern classification. If a RAM cell finds its assigned feature then it outputs a 1. The sum of these outputs gives the response, r of that discriminator; the closer the value of r is to K the more certain the match. To perform character recognition, 26 discriminators are required, each of which is trained to identify a particular letter from a binary bit map. The responses of the discriminators for a given input pattern are compared with the highest responding discriminator identifying the character which is best match to the input pattern. Due to the use of RAM this style of system is very fast but the implementation is very specific to pattern recognition and cannot be viewed as a general purpose neural computing engine.

3.2.4. Data Flow Architectures

The architectures of digital VLSI implementations fall into two categories

1. Broadcast Data Flow (Figure 3.5)
2. Systolic Data Flow (Figure 3.6)

In a system based on a broadcast architecture one common/global databus links all of the processing nodes (PN) together. This flexible structure in which any 2 PN's can

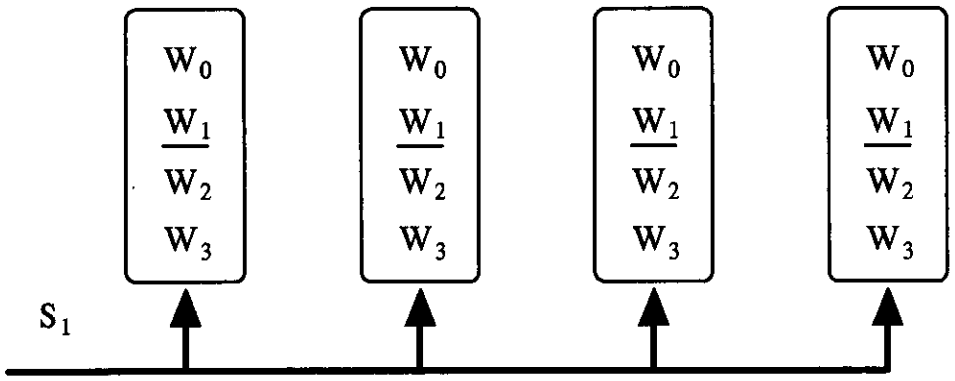


Figure 3.5 Broadcast Data Flow.

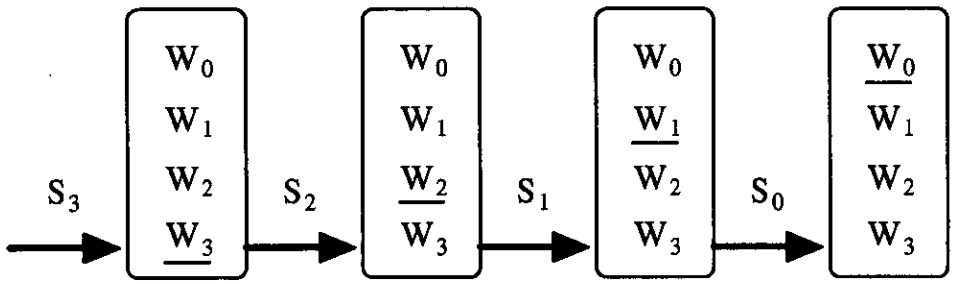


Figure 3.6 Systolic Data Flow.

communicate with each other, is well suited to fully interconnected networks such as Hopfield/Tank. However, it takes a long time to transfer data between elements as each PN has to access the bus in turn to output its results. A transmission overhead also exists. With every piece of data broadcast, an address header is required to determine which PN is sending data.

A broadcast databus forms the backbone of Hitachi's Wafer Scale Integrated neural network [50]. It not only transfers data between the 12 neurons within each integrated circuit but also links the 49 chips contained in the wafer. In this case a 10-bit address is sent with each 9-bit neural state. The resulting cycle time to fully interconnect the 576 neurons within the wafer is 267μs which allows the system to solve the computationally hard, 16-city Travelling Salesman Problem using the Hopfield/Tank neural network in an impressive 0.1s.

The Connected Network of Adaptive ProcessorS (CNAPS) by Adaptive Solutions Inc combines this broadcast architecture with a cascadable linear array of 64 "DSP-like" 16-bit fixed point processors [51]. Even with the use of 0.8μm technology the integrated

circuit which contains these 64 PN's is still 26.2mm by 27.5mm in size. To deal with the problems caused by the fabrication defects that are almost inevitable in such a large die, redundancy has been built in.

At the other end of the scale from the Hitachi and CNAPS VLSI implementations is the TInMANN integrated circuit by Melton *et al* [52, 53]. As the designers were limited to less than $4mm^2$ they were only able to include one neuron on the chip. The chip was thus designed as a building block with the broadcast architecture used to allow the chip to be easily cascaded. A feature of this particular implementation is that the neuron structure has been tailored for the integer Markovian learning algorithm which only requires adders and subtractors.

Common to these three implementations is a neuron plus its associated synaptic column implemented as a single processing node which processes each of the synaptic weights contained in a local block of RAM in turn. This scales very well, as only the RAM size has to be increased to allow more weights to be down-loaded to the accelerator board. The CNAPS device has 4Kbytes of cache RAM at each PN to store that neuron's synaptic weight values. For image processing applications, where the weights are constant once training is complete, this cache RAM gives a significant speed advantage as the weights do not have to be reloaded for each calculation.

The systolic implementations proposed by Jones *et al* [54, 55] and Mackie [56, 57] are similar to the TInMANN design in that the PNs also have local weight RAM. In a systolic architecture the PNs are "daisy-chained" together like D-Type flips-flops in a shift-register. After every multiplication-accumulate cycle the data is passed on to the next PN in the chain. Thus the data moves on one PN per multiplication cycle; if there are N PN's in the array it will therefore take N cycles for the data to be received by all the PN's. For linear arrays the data flow is along the array and for 2 dimensional array there are typically 2 dataflows one across the array and a second down the array. The local and regular nature of interconnect makes this architecture particularly suitable for VLSI implementations.

The toroidal neural network developed by Jones *et al* [54, 55] is a good example of a systolic architecture (Figure 3.7). The toroidal structure derives from the presence of a systolic ring, around which the neural states circulate, and the similarity of the transfer of weight information within the PN, to data moving around a loop. The movement of the weight data is synchronised with the movement of the neural state systolic ring so that the correct neural state and synaptic weight coincide at the required PN. Each PN comprises a 16 by 16-bit fixed point multiplier and a 16-bit adder/subtractor. Also included are eighteen 16-bit registers for storing data, a linear feedback shift register for random number generation and a comparator.

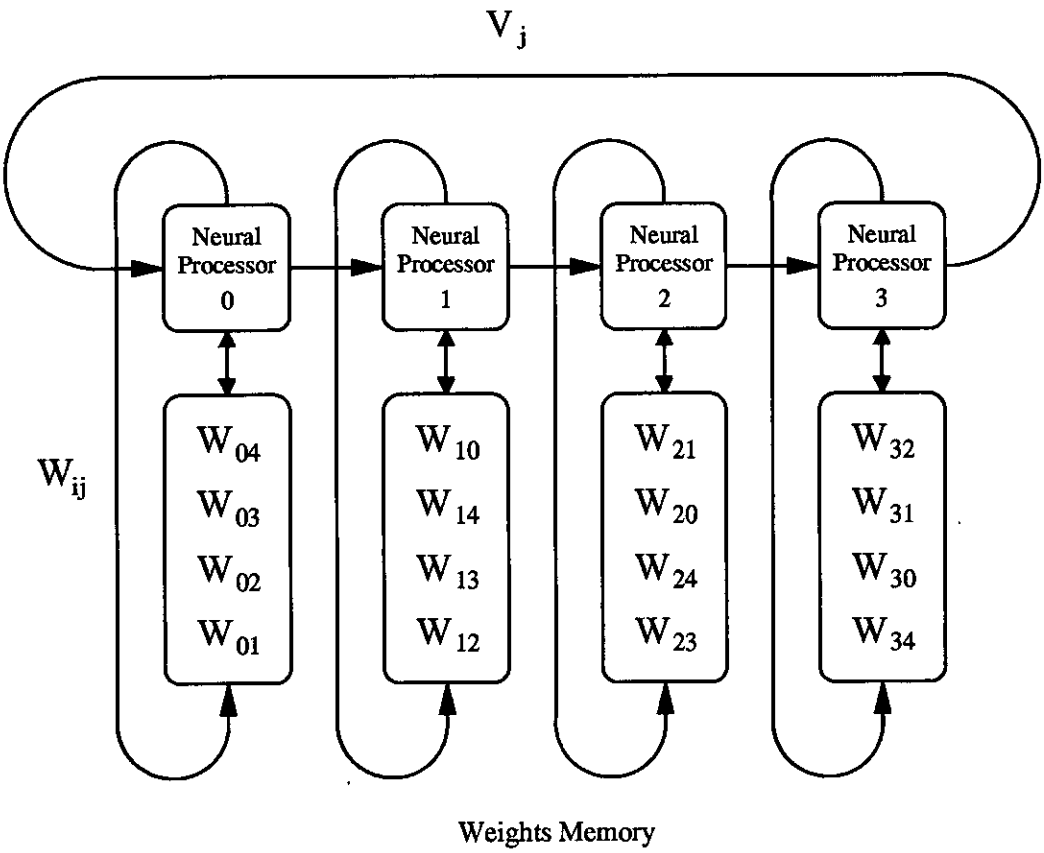


Figure 3.7 A Toroidal Neural Network.

The work by Mackie [56, 57] is based on a linear systolic array of bit-serial processing elements optimised for pattern recognition problems. The optimisations are based on the premise that in these applications the weight data remains static once trained; thus the resulting pipelined architecture concentrates on a fast throughput of input vectors (neural states).

The systolic Neural Signal Processor (NSP), the MA16, developed by Ramacher *et al* [58, 59] differs from the work of Jones and Mackie in that it performs matrix-matrix multiplication rather than vector-matrix multiplication. The NSP chip contains 4 PN's, each of which consists of four 16x16 bit multipliers and three adders linked in a systolic manner. Each PN computes the result for a 4x4 submatrix of the overall matrix-matrix multiplication. As Figure 3.8 shows, the 2 dimensional flow of data needs to be carefully ordered if the correct data is to reach the appropriate processor at the correct time especially when the matrix is divided over several processors. The core area of MA16 covers 154mm^2 of silicon and contains over 500K transistors. Clocking the device at 50MHz achieves a throughput of 800 MCPS

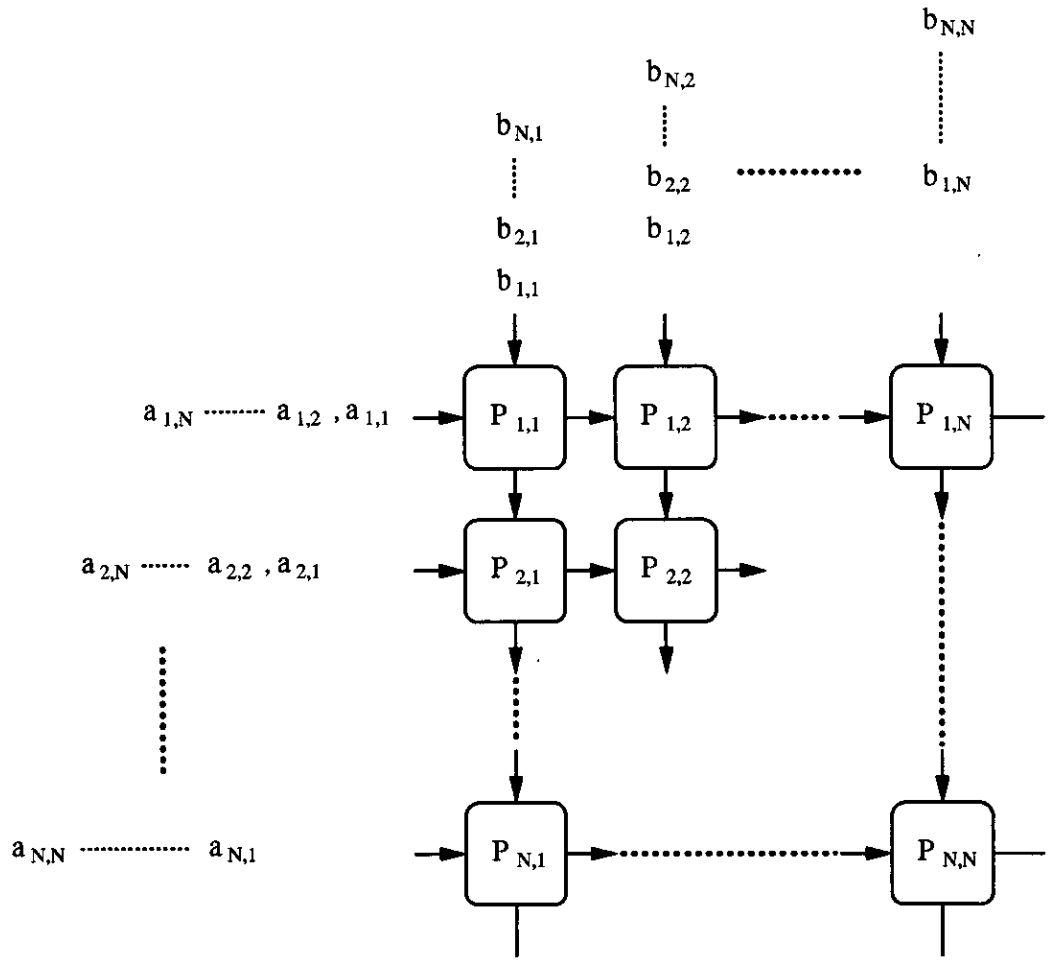


Figure 3.8 A Systolic Matrix Multiplication Scheme.

3.2.5. Conclusions

Table 3.1 summarises the main advantages and disadvantages of digital techniques. The use of digital techniques eases the interfacing of a neural system to a host computer as there is no need to convert the data into voltages, as in an analogue implementation, before it can be processed. Thus for a neural network hardware accelerator, a digital implementation is probably the most sensible route to follow.

3.3. Analogue VLSI Implementations

The Hopfield/Tank neural network model is derived from an analogue implementation based on resistors, capacitors and operational amplifiers. The resistors represent synapses, while the currents they carry are equal to the multiplication of the voltage across them (the neural state) by the reciprocal of their resistance (the synaptic weight). However as resistors are not well suited to CMOS VLSI implementations a more

Advantages	Disadvantages
Fast	Large Area
Reliable	High Power Consumption
Good Noise Immunity	Small speed/area product
Arbitrary Accuracy	
Familiar Design Technology	

Table 3.1 Summary of the Advantages and Disadvantages of Digital Multipliers.

intelligent approach is needed.

As the multiply accumulate operations are common to both neural networks and analogue filters and correlators, many of the circuits used in analogue VLSI neural implementations have their roots in analogue integrated filter design work. Indeed many implementations can be viewed as general-purpose analogue vector-matrix multipliers rather than dedicated neural processors.

The basic multiplier cell used in analogue VLSI neural implementations can generally be classified into one of five different categories. The fixed value nature of VLSI resistors excludes them from this classification as any system based around them would be non-programmable.

1 Gilbert Multiplier.

2 Sub-Threshold Multipliers.

3 Linear Transconductance Multipliers.

4 Multiplying Digital-to-Analogue Converters (MDAC's).

5 Switched Current Sources.

The common factor is that the synapse output always takes the form of a current. Kirchhoff's Current Law states that all currents at a circuit node must sum to zero. Thus addition of the synapse currents is elegantly and simply achieved by connecting the outputs to a common bus. The remainder of this section discusses implementations based on these multiplier techniques and methods for storing analogue weights.

3.3.1. Gilbert Multiplier

The Gilbert Multiplier was developed by Barrie Gilbert in 1968 for bipolar circuits. The CMOS variants of this 2-input, 4-quadrant analogue multiplier use almost identical circuits. As Figure 3.9 shows, the CMOS version of a Gilbert multiplier is built up of 3 differential stages which steer current between the 2 transistors in each stage according to

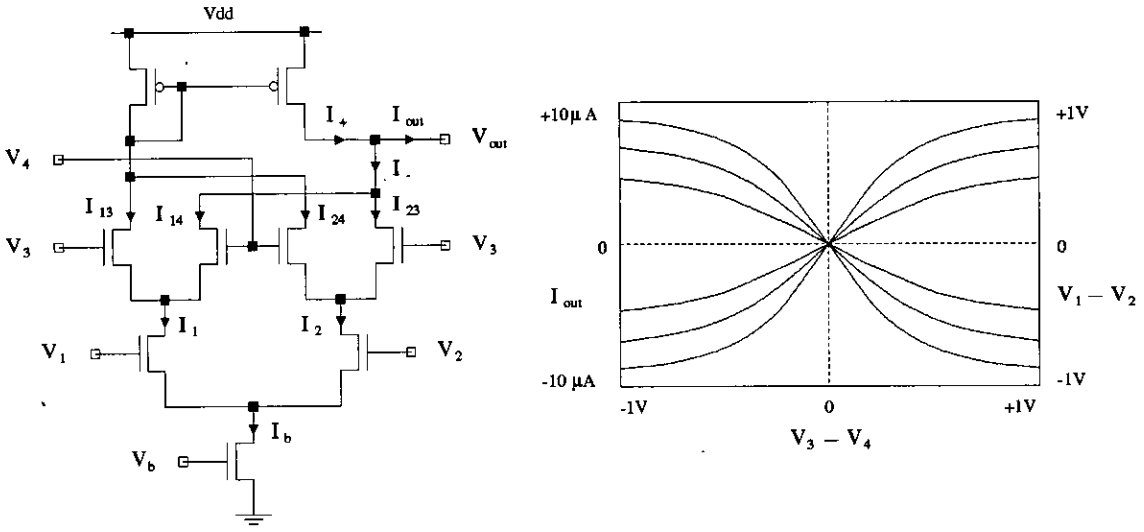


Figure 3.9 Circuit Diagram for a Gilbert Multiplier

the difference between the gate-source voltages of the transistors. The inputs to the circuit are the voltage differences $(V_1 - V_2)$ and $(V_3 - V_4)$, with the output current, I_{out}, proportional to the multiplication of these 2 voltage differences. The summation of the synaptic multiplies is achieved simply by connecting the current outputs to a common bus which feeds into a neuron. As the multiplier characteristic in Figure 3.9 shows, the response is linear for small voltage differences with the output saturating for larger differences due to one leg of a differential stage winning all of the available tail current. Thus for linear multiplication the input range needs to be restrained. The overall tanh shape of the response however can be useful for implementing the sigmoidal squashing function required by networks.

If V_{out} falls below the maximum value of V_3 or V_4 then the transistors in the top 2 differential stages will no longer be in their saturation regions, making I_{out} dependent on the value of V_{out}. Thus a limitation of the circuit as it stands is that the output current, I_{out} is only valid for a limited range of V_{out} values. Not only is the range of input voltage differences restricted, but the lowest values for V_3 or V_4 must be at least a threshold voltage above the highest values of V_1 or V_2, otherwise the transistors in the top 2 differential stages will switch off. As the successful VLSI implementations by Intel [60, 61] and Kub *et al* [62] show these problems are of only minor significance.

The Electrically Trainable Artificial Neural Network (ETANN) chip from Intel contains 10240 synapses based on this version of the Gilbert multiplier. The synapses are organised into two 80 by 64 arrays which feed into 64 variable gain neurons. The voltage difference representing the synaptic weight is held by a pair of EEPROM cells in the

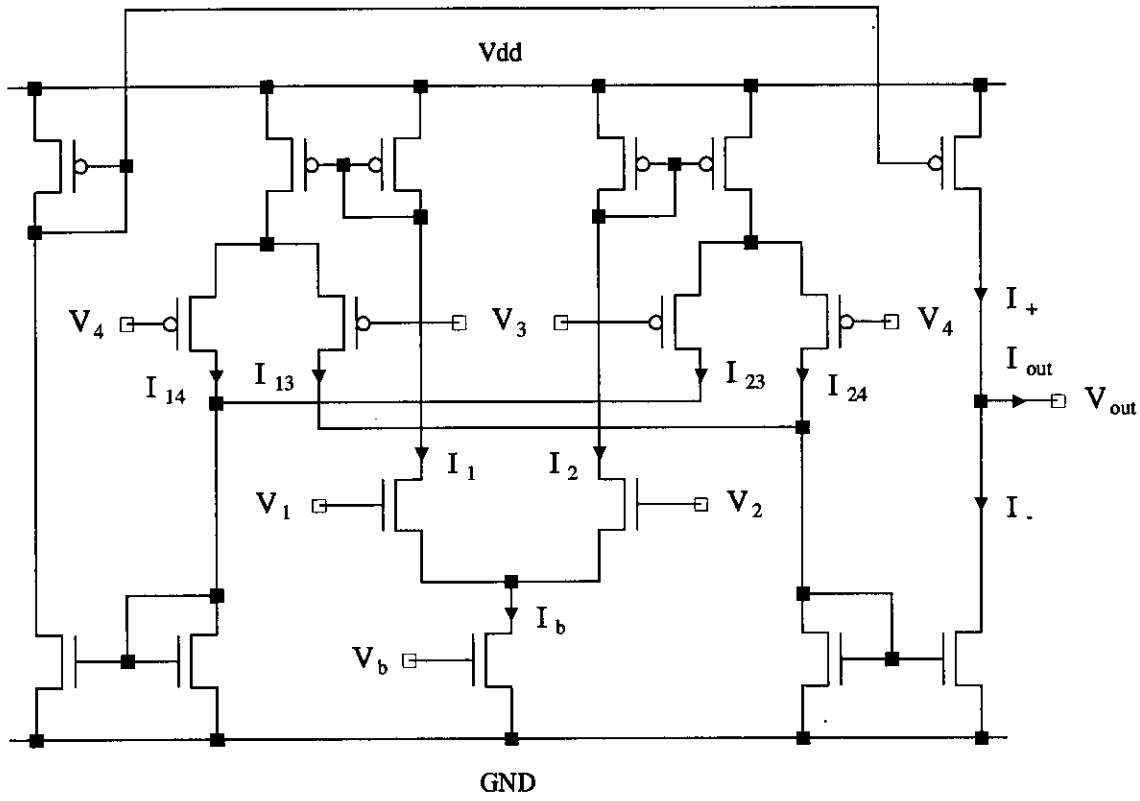


Figure 3.10 Circuit Diagram for the Wide-Range Gilbert Multiplier

synapse. The advantage of EEPROM technology is that weights are non-volatile, but to change a single weight takes 100's of microseconds, implying slow weight update times during the learning phase of a MLP. Once learning is complete and the weights are constant, the device is capable of 2,000 MCPS. Intel market a board that allows 8 ETANN's to be connected together but the problems of process variation between chips gives such a system a rather non-uniform synaptic multiplier characteristic. This results in a significant degradation of the network's performance.

The work by Kub *et al* [62] at the Naval Research Laboratory in Washington DC is an example of the Gilbert multiplier being applied to a general purpose analogue vector-matrix multiplier. Each weight in the 32 by 32 synaptic array fabricated, is stored as a voltage difference between 2 capacitors. The assumption is that, as the capacitors decay at the same rate, the voltage difference between then will remain roughly constant, extending the weight hold times by a factor of 50.

The wide-range version of the Gilbert multiplier (Figure 3.10) uses current mirrors to direct currents between the differential stages and on to the output node. This avoids the problems of V_{out} affecting I_{out} and V_1, V_2 and V_3, V_4 needing to vary about different voltages for correct operation. The only drawback is increased silicon area due to a

doubling in the transistor count. As a major objective of VLSI implementations is to integrate as many synapses as is possible on to a single die, the more compact version of the Gilbert multiplier has obvious attractions. Nevertheless, the wide-band variant has been used by Schneider and Card [63] in their VLSI implementation of Hebbian and Mean Field Learning (See Section 3.4.3)

Although the multiplier used by Satyanarayana *et al* [64, 65] in their reconfigurable neural network chip is not explicitly described as a wide-range Gilbert multiplier, the circuit layout given is very similar. One of the features of this chip is the use of a distributed neuron. Each synapse has two simple current-to-voltage converters to translate the differential current output from the Gilbert multiplier into a differential voltage and apply a sigmoidal squashing function. When an array of these cells is cascaded together to form a synaptic column, the I to V stages operate in parallel to create a much bigger output stage; thus this system is cascadable. The reconfigurability is due to the distributed synapse/neurons being arranged in 4 by 4 blocks, with the links between adjacent blocks programmable to allow defective synapse/neurons to be isolated and the chip to be configured to suit the topology of the proposed application. The Satyanarayana chip contains 1024 of these synapse/neuron cells arranged into 4 by 4 blocks. Of the chip's 6.7mm by 3.4mm area, 15% is occupied by the switches for re-configuring the interconnections between blocks.

3.3.2. Sub-Threshold Multipliers

In sub-threshold circuits the gate-source voltage (V_{GS}) of a transistor is less than its threshold voltage (V_T). For this region of operation the transistor current has an exponential relationship to the value of V_{GS}. The output current saturates for V_{DS} values as low as 100mV, allowing the transistor to operate as a current source from near ground to V_{dd}. As the transistor is not fully switched on the currents are very small (10^{-12} to 10^{-6}A). Thus the power consumption of circuits biased in the sub-threshold regions is very small allowing large numbers of circuits to be integrated on to a single die without incurring power dissipation problems. The small value of V_{GS} also enables the circuits to work from low power supply voltages eg 1.5V. A disadvantage is that input voltage ranges are limited to 100's of mV creating the potential for noise problems.

The Gilbert multiplier in (Figure 3.9) works well with the transistors in their sub-threshold region of operation. The differential stages in the multiplier act to cancel out the exponential nature of the transistors' response, yielding an overall tanh transfer characteristic. This is almost identical to the response given by the Gilbert multiplier when the transistors are in their more common saturation region. To achieve sub-threshold operation the input voltages must obey the restriction that $V_b > V_1$ and $V_2 > (V_3$ and $V_4)$, otherwise the transistors will operate in their saturation regions.

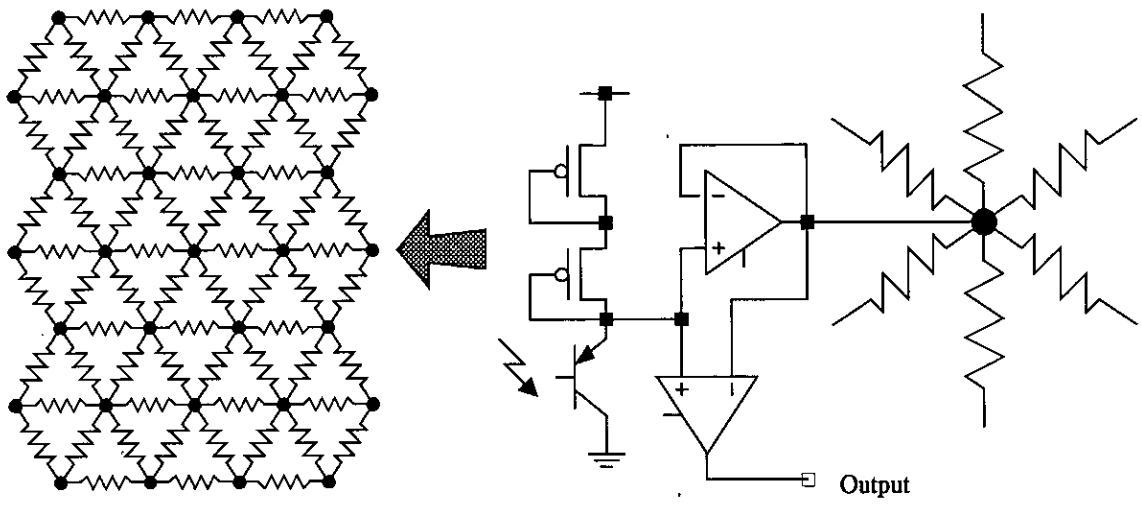


Figure 3.11 Silicon Retina with Schematic of Mahowald Pixel.

Carver Mead's research group at Caltech have used sub-threshold circuits to implement silicon retina and electronic cochlea chips inspired by the biological exemplars [66-69]. The silicon retina chip is described below to illustrate the sub-threshold work of Mead *et al.*

The 2 dimensional structure of a human retina translates well into silicon. As Figure 3.11 shows, the retina chip uses a hexagonal resistive grid structure with each node consisting of a photoreceptor and two amplifiers. The voltages at nodes in such a structure represent a spatially weighted average of the photoreceptor outputs. The more distant an input is from a point, the smaller is its weight (effect). Thus each input has a significant effect only on the nodes around it.

The photoreceptor uses sub-threshold load transistors to compress logarithmically the order of magnitude variations in current produced by the photo-transistor into a manageable voltage range. One of the amplifiers then tries to drive the associated resistive node towards the photoreceptor voltage. The voltage difference between the locally-averaged grid node voltage and the photoreceptor voltage is a measure of the contrast ratio or intensity gradient at that point, as the voltages represent the logarithms of the intensity values.

Working on local averages gives both the human eye and this silicon retina the ability to find detail in both light and dark areas on the same image. As the output also represents the intensity gradient, a large output will indicate that a transition or edge is present at that point in the image. This is similar in concept to the Laplacian filters used in present day computer vision systems.

The main advantages of operating in the sub-threshold region are the low power consumption which is important for large synaptic arrays and the ability to operate from power supply voltages as low as 1.5V (battery power). Also the logarithmic transfer function simplifies synaptic multiplication to the addition of currents. A problem with this mode of operation is that the limited input voltage ranges mean that the circuits are susceptible to noise. The small currents flowing in the circuits result in a limited drive capability and a slow response to a change in input data.

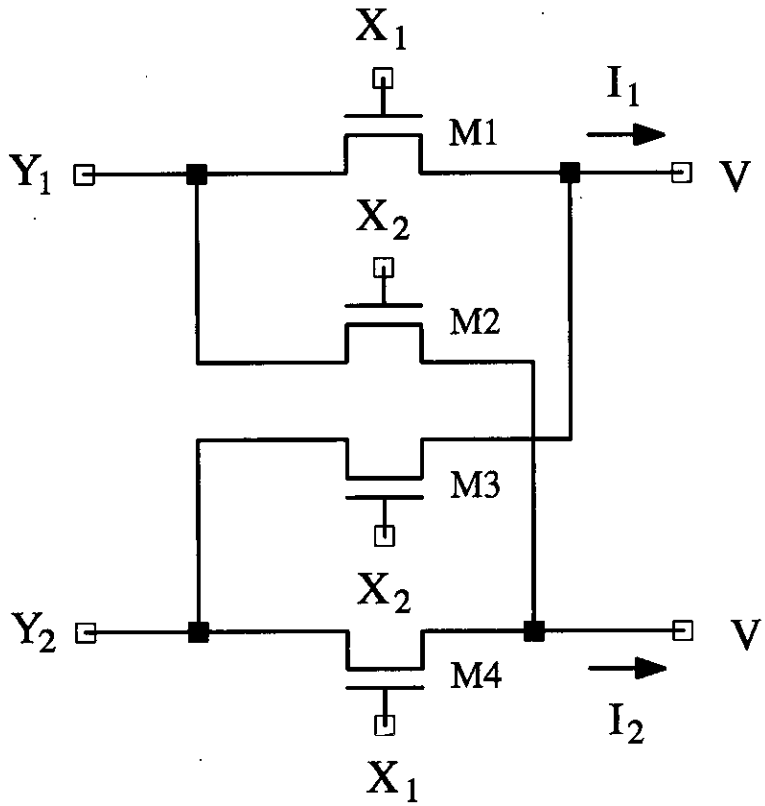


Figure 3.12 4 Transistor MOS Transconductor

3.3.3. Linear Transconductance Multipliers

The current through a NMOS transistor in its linear region of operation is roughly proportional to the multiplication of its gate-source voltage, V_{GS}, and drain-source voltage, V_{DS}. This characteristic is the basis of the highly linear multipliers used in many MOS integrated filter designs [70-75]. In general, the designs are based on the 4-quadrant transconductance multiplier shown in Figure 3.12. The response of the 4 identical transistors in this circuit is

$$I_1 - I_2 = \beta\left[(Y_1 - Y_2)(X_1 - X_2)\right] \tag{3.1}$$

where β is the transistors' transconductance, confirming the linear nature of the circuit's 4-quadrant multiplication.

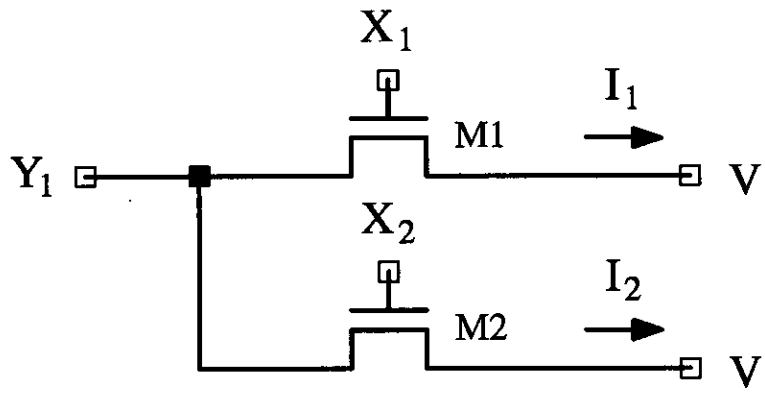


Figure 3.13 2 Transistor MOS Transconductor

Figure 3.13 illustrates a 4-quadrant multiplier based on just 2 transistors in their linear regions. The response of the 2 transistor version is

$$I_1 - I_2 = \beta\left[(Y_1 - V)(X_1 - X_2)\right] \tag{3.2}$$

As Equation 3.2 shows, the 2 transistor multiplier suffers from the disadvantage that its output current difference is dependent on the value of reference voltage V. Thus the 4 transistor variant is more immune to variations in V.

Tsividis and Satyanarayana [76], Bibyk and Ismail [77], Verleysen, Jespers *et al* [78, 79] have all proposed the use of linear transconductance multipliers for neural networks but no-one has yet reported results from a neural network implementation based on these circuits. A variant of the 2 transistor multiplier is however used as a linear, voltage controlled current source in our own more recent pulse stream work [80].

3.3.4. Multiplying Digital to Analogue Converters (MDACs)

Unlike the previous three implementation styles where both the neural state and synaptic weight information are represented as voltages, in a MDAC one of the inputs is digital. This is an example of a hybrid implementation. In an N-bit CMOS DAC implementation based on transistors there are N switchable current sources, the magnitudes of which increase by a factor of 2 as the significance of the associated bit increases. The magnitude of the output currents is controlled by a common current mirror whose input is I_{Ref}, and the ratioing of current mirrors is achieved by doubling the aspect ratio of the current source transistors as the bit significance increases. The output current is thus proportional to I_{Ref} multiplied by the number represented by the N-bit digital input. The

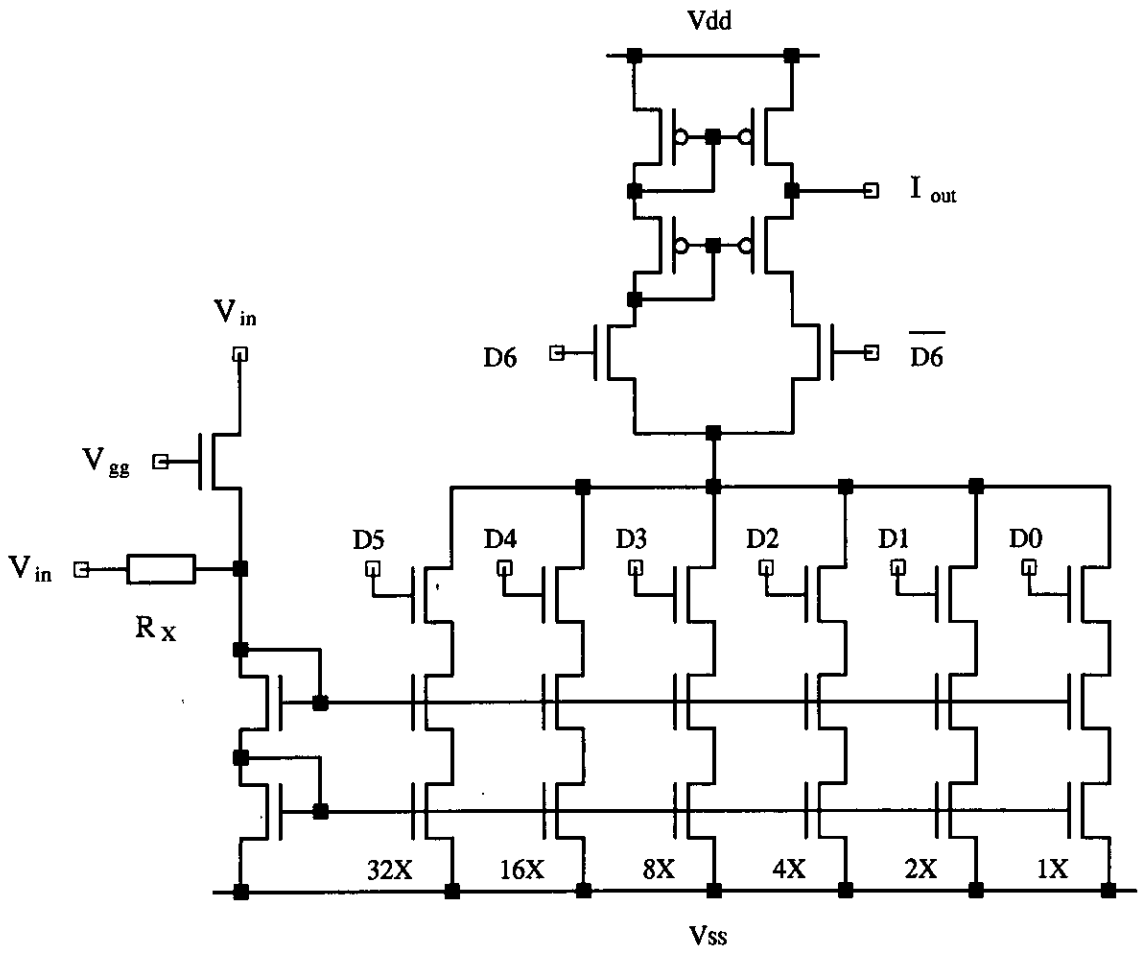


Figure 3.14 Circuit Diagram of 7-bit Multiplying DAC.

advantage of this particular implementational style is that the interface to a digital host computer is now more straightforward. However as one of the inputs to the network is quantised typically to 7-bit accuracy there may be problems during the learning phase of an MLP network.

In the MDAC implementation (Figure 3.14) by Moopenn *et al* [81] the digital input represents the weight while either a transistor or an external resistor converts the input neural state voltage into I_{Ref} for the appropriate MDAC's. As Figure 3.14 shows the digital input is 7-bits with the most significant bit used as a sign bit to reverse the direction of the output current via a differential stage. Only 2-quadrant multiplication is possible using this particular arrangement. The fabricated device contains a 32 by 32 array of multiplier cells. An array of four of these chips has been used successfully to solve the 8-city Hopfield/Tank TSP in about 50μs, producing the optimal tour 11% of the time.

The work by Hollis and Paulos [82] is similar, with the weights stored as 7-bit digital values but the neural state is represented as a voltage difference. Multiplication is

achieved in this implementation, not by varying the reference current I_{Ref}, but rather by a differential stage multiplying the DAC output current by a factor less than 1 determined by the neural state voltage difference. For negative weights the sign bit of the weight reverses the polarity of the differential input. Load transistors convert the differential output current into a differential voltage. A transistor connected between the 2 current summation lines provides a primitive gain control with the output zero when the transistor is on and maximum gain when it is off. As the proposed application is a fully interconnected network like Hopfield/Tank, the input voltage levels need to be the same as the output voltage levels. To ensure this a common feedback circuit controls the value of a bias current to give the desired levels for the output voltage range. The fabricated integrated circuit contains 7 neurons fully interconnected via an array of 49 MDAC synapses. The neuron is as used by Sivilotti *et al* [83]. It has successfully solved small scale graph partitioning problems.

The Analogue Neural Network Arithmetic (ANNA) unit is a well established VLSI neural implementation based on MDAC circuitry [84-86]. Despite weights inside the device being represented by voltages, the interface to the chip is purely digital with two on-chip DAC's converting the 6-bit digital weight values into the appropriate voltages. The neural states are limited to 3-bit accuracy. The pattern recognition results reported indicate that this limited accuracy is adequate except in the final layer of an MLP network. However this will restrict the range of applications which will run successfully on this hardware. The adverse effects of quantisation of values on network performance also apply to the Moopenn and Hollis implementation. On the system board for the ANNA chip a floating point DSP-32C processor is provided to calculate the final layer of an MLP network and perform learning, partially alleviating the quantisation problem. The ANNA chip comprises 4096 synapses and 8 linear neurons, and can handle up to 256 neural state inputs. The aspect ratio of the synapse array is re-configurable, supporting neurons with 16 to 256 inputs. The silicon area of the die is 4.5mm by 7mm (0.9μm CMOS process). Peak computation rate of the ANNA chip is 5000 MCPS. The average performance of the system as a whole much less at about 1000 to 2000MCPS. This is because the supporting architecture is not able to supply data to the multipliers inside ANNA fast enough, to maintain the multipliers at their maximum computation rate.

3.3.5. Switched Current Sources

In some neural applications it is feasible to binarise the neural state values to either 0 or 1. This simplifies the synapse circuitry to a variable current source and a switch to gate its output. Ismail *et al* [77, 87] use a differential stage made up of 2 floating-gate MOS (FGMOS) transistors, M1 and M2, and a current source, M3 (Figure 3.15) to create a voltage controlled current source. The difference between M1 and M2's pre-

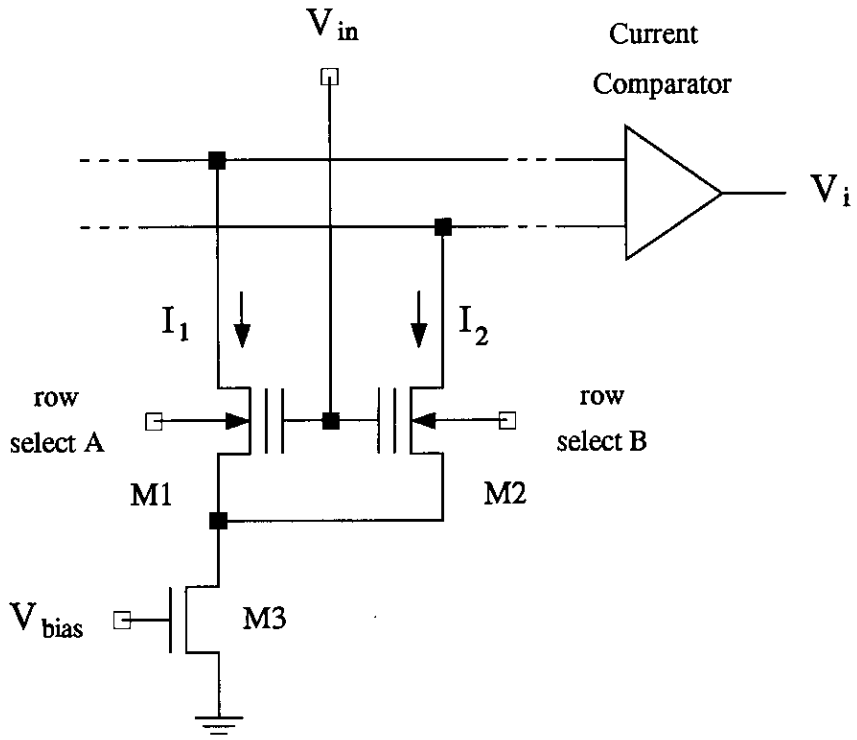


Figure 3.15 Ismail *et al* Programmable Synaptic Element (PSE).

programmed threshold voltages determines the difference between currents I_1 and I_2. The threshold voltage difference represents the synaptic weight in an analogue manner. V_{in}, which is either ground or V_{dd}, switches the FGMOS transistors off or on, so gating the output current difference. A testchip, containing a 4 by 4 array of these cells and current comparators for the neurons, successfully proved the operation of this circuit.

The implementations by Verleysen [88, 89] and Graf [90] simplify the synapse further by restricting the synaptic weight range to {-1.0,0,+1.0}. In these systems the neural states are limited to {-1.0,+1.0} rather than {0,+1.0}. An XOR gate is used in the synapse to re-direct the fixed-value current source between the positive, I_+, and negative, I_-, current summation lines (Figure 3.16 (a)). To turn the synapse off, Mem 1, the control voltage for the current source, is set to 0. As Figure 3.16 (b) shows, load transistors convert the summed current difference into a voltage difference. A comparator then translates this into the binary neural output. Because of the simplifications made to the synapse a new learning algorithm which took into account the restrictions on the synapse weights and the neural state had to be developed.

The 32000 synapses and 256 neurons contained in the reconfigurable neural network chip developed at At&T Bell Laboratories by Graf *et al* demonstrates the high levels of integration which can be obtained by the use of binary weights and states. To run

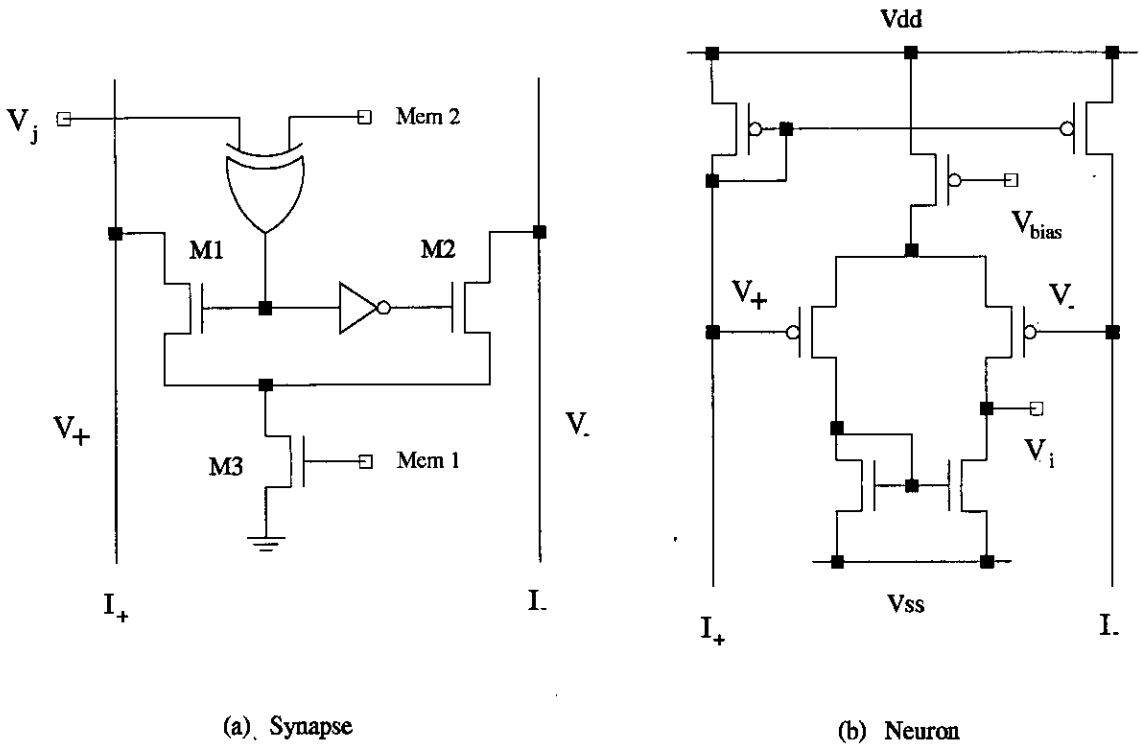


Figure 3.16 Verleysen *et al* synapse and neuron circuits.

networks that require a higher level of synaptic precision, up to 4 synaptic columns can be cascaded together to form a 4-bit weight. While this increases the precision of the system it reduces the number of synapses available. Thus there is a direct trade off between the accuracy of the weights and states, and the silicon area required. A multiplier in each neuron scales each column by the appropriate factor {1,1/2,1/4,1/8}. Two internal 128-bit wide shift registers are present to speed-up the computation throughput for signals or images which are network scans. The speed of the bit-wise computation is very fast with the chip rated at 320,000MCPS. The pattern classification results reported demonstrate that a system based on such simple processing elements is capable of performing feature extraction and convolutional algorithms very effectively, though obviously the limited computational accuracy will limit the range of applications.

3.3.6. Analogue Weight Storage Techniques

Normally in analogue VLSI neural implementations, the synaptic weight values are stored locally in the synapse. Thus a way of storing the analogue voltage which represents the weight value is required.

The ideal analogue storage mechanism needs to be small, to allow high levels of integration, and continuous in nature. The desire for on-chip learning also means that it

must be easily adjustable and non-volatile [91]. There presently exist 4 main techniques for programmable analogue weight storage. The programmability requirement precludes VLSI resistors from the following discussion as their value is fixed at the time of fabrication.

1 Capacitors

2 Floating-gate MOS (FGMOS) Transistors

3 Charge-Coupled Devices (CCDs)

4 Amorphous Silicon

None of these techniques satisfies all the criteria for an ideal analogue storage mechanism. The following sections outline the merits and demerits of each technique.

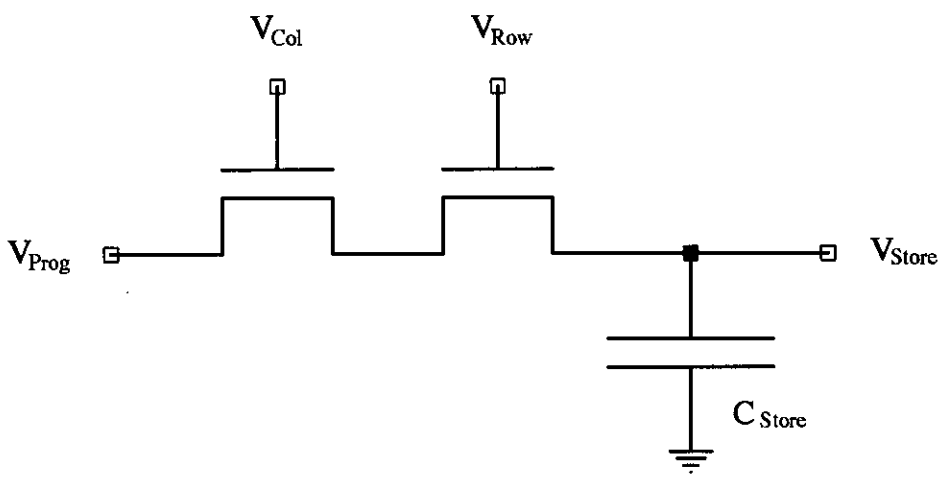


Figure 3.17 A simple capacitive sample-hold.

3.3.6.1. Capacitive Weight Storage

The most common way to store a synaptic weight voltage is to use a simple sample-hold circuit based on a capacitor (Figure 3.17). Its advantages are that it is compact, easy to increment/decrement via charge pump circuitry, and continuous. The problem with this technique is that charge leakage through the access transistors causes the stored voltage to decay in time. As the access transistors, when they are off, are in their sub-threshold regions, the leakage current is of the order 10^{-10} to 10^{-12}A. For these current values, the weight storage capacitor will hold the weight voltage to within 1% of the maximum weight voltage for seconds. Thus the weight voltages need to be continuously refreshed, typically from external RAM via an external DAC. The refresh operation is normally transparent to the neural network application running on the chip. Care does need to be exercised as the RAM/DAC combination introduces quantisation effects into

the weights.

The Satyanarayana *et al* [65] chip uses such an arrangement and a single external DAC is able to update a weight in 1μs (1 MHz). To reduce the time taken to load the weights this chip has 8 input weight lines, which when driven in parallel allow the 1024 synapses to be updated in 130μs. The ANNA chip [84] achieves a much faster weight update time of 50ns (20 MHz) per weight through the use of twin on-chip DAC's and current-mode operation. The twin DAC's are able to refresh all of the 4096 synapses in 110μs. Thus not only does a weight refresh scheme eliminate the weight decay problem, it allows the complete weight array to be updated very quickly.

To extend the weight storage time, the synaptic weight can be stored as a voltage difference between two capacitors. The idea is that the voltage on each capacitor decays at the same rate, so the voltage difference will be preserved for a much longer time. The work by Kub *et al* [62] shows that the storage of a voltage difference reduces the weight decay rate compared to that of a single capacitor by a factor of 50 from 30mV/s to 0.6mV/s at room temperature.

The flexibility and update speed possible with capacitive storage are reasons why at present, it is the most common analogue weight storage technique.

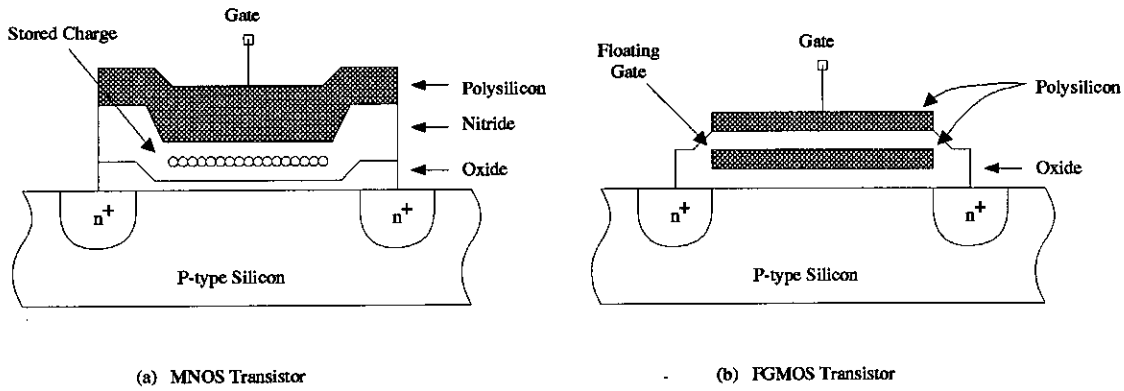


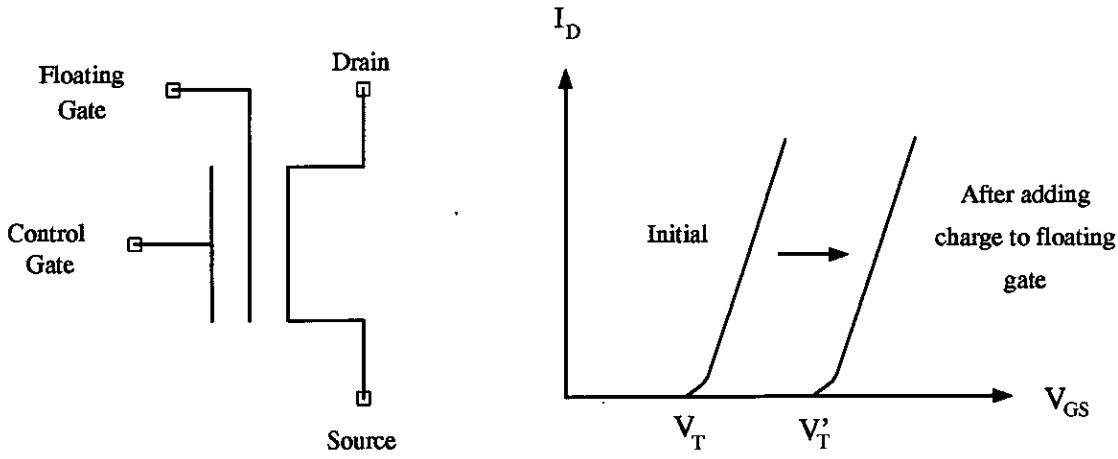
Figure 3.18 Cross-sections of MNOS and FGMOS transistors.

3.3.6.2. MNOS and FGMOS Transistors

The Metal-Nitride-Oxide-Semiconductor (MNOS) and Floating-Gate MOS (FGMOS) transistors are examples of programmable threshold-voltage transistors. They are widely used in electrically erasable programmable read-only memories (EEPROM) and electrically programmable ROM's (EPROM) respectively, to provide non-volatile storage of digital data.

For both devices, the threshold voltage is altered by modifying the size of charge layer between the transistor's gate and drain-source channel (Figure 3.18). This in turn alters the gate voltage required to create the depletion region between the source and drain, and so turn the transistor on. By varying the amount of charge stored, the transistor's threshold voltage can be made to change in an analogue manner, creating a non-volatile, analogue memory cell. As Figure 3.19 shows, adding charge to a floating gate is equivalent to adding a positive offset to the transistor's characteristic.

For the MNOS transistor large voltage pulses (±10V) are used to tunnel charge into the nitride layer between the polysilicon gate and the silicon oxide layer. By changing the polarity of the pulse the threshold voltage can either be incremented or decremented. Programming is an iterative process, with the transistor's threshold voltage examined after each cycle to determine if another pulse is needed to increment or decrement the stored charge. The programming pulses are typically 100μs in length so setting one threshold voltage may take 100's of μs. The result is weight load times which are 100 to 1000 times slower than capacitive weight storage. On the other hand the transistor has the ability to retain digital information for 1 to 10 years depending on device. Intel's ETANN chip [61] uses two MNOS transistors to define the synaptic weight as a difference in threshold voltages. The relatively slow update times for a MNOS device mean that ETANN becomes very slow during the learning phase of the network.



(a) Symbol (b) Characteristic

Figure 3.19 The MNOS/FGMOS Transistor Symbol and Characteristic.

Rather than storing charge in an insulating layer, a FGMOS transistor stores charge in a thin conducting layer trapped between two insulation layers. The FGMOS transistors in EPROM's use avalanche breakdown to store charge on the floating-gate; however

radiation with ultra-violet light is required to remove the stored charge. The FGMOS transistor in Figure 3.18(b) overcomes this requirement by having a polysilicon top gate to which a negative voltage is applied. This encourages hole injection into the floating gate, so reducing the charge present. Its behaviour is now similar to that of a MNOS device. Only Borgstrom *et al* [87] have so far proposed the use of this particular programmable threshold transistor.

Both the MNOS and FGMOS transistors outlined require specialist fabrication processes due to the additional processing steps need to create the charge storage locations.

3.3.6.3. Charge-Coupled Devices

In charge-coupled devices (CCD) information is passed as packets of charge instead of as a current or a voltage. By using a three phase clock signal a CCD shift register can be constructed. The main application of such shift registers is to read out the charge generated by photo detectors as part of an imager array. The CCD cell has the advantage of being very compact.

As size of the charge packet is an analogue quantity it can be used to represent a synaptic weight. Agranat *et al* [92] have fabricated a CCD based neural integrated circuit in which the charge packets representing the weights can be loaded either electrically or optically in a manner similar to a CCD camera. The use of an optical weight input was an attempt to speed up the weight load times which are presently one of main bottlenecks in VLSI neural implementations. However, the electrical weight inputs were found to be more reliable and consistent. Each of the 256 neurons consists of an integrator and a CCD ring of 256 shift registers. Neural states are either 0 or 1, so either the weight charge packet is reproduced and transferred into the integrator (a very large CCD cell) or it is ignored.

A successor to this chip replaced the 256 element shift registers with an array of synapses each consisting of 2 CCD cells [93]. One cell contains the synaptic weight charge; if the binary state input is 1 then during the computation phase the charge packet will be transferred into the adjacent cell. As the top electrodes for the synapse output cells within a synaptic column are commoned together, the capacitances of the CCD cells add together. Thus the voltage of this common electrode is the sum of the charge packets transferred divided by the total capacitance of the electrode. The neuron has the ability to divide its output voltage by 2 so that the system can process multi-bit neural states in a bit-serial fashion. To emphasise the compact nature of CCD, the synapse just outlined only occupies 24μm by 24μm.

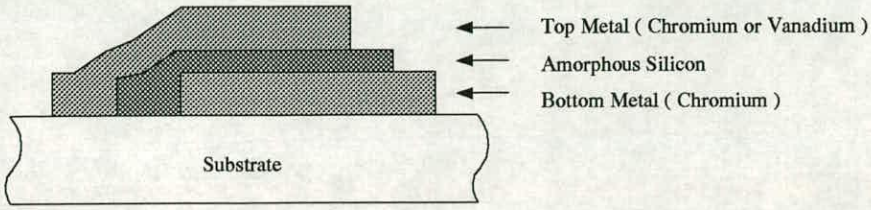


Figure 3.20 Cross-section of an Amorphous Silicon Programmable Resistor.

3.3.6.4. Amorphous Silicon

A three layer sandwich of metal, amorphous silicon and metal forms a two terminal device the resistance of which can be programmed using variable height voltages pulses (Figure 3.20). The first devices developed by Hajto *et al* [94] used a chromium-amorphous-chromium structure. This gave a digital device which had two stable resistance states, 10^3 and 10^6 ohms. Analogue devices were subsequently created by replacing the chromium top metal with vanadium. The device size is typically 10μm by 10μm. By varying the height of the 100ns programming pulses between ±1V and ±4V the resistor can be set to any value in the 10^3 to 10^6 ohm range. As with MNOS transistors, programming is an iterative affair with several attempts normally required before the device has the correct resistance. Due to the use of 100ns pulses the process of setting the resistances is much faster than with MNOS devices. The devices have two main problems.

1 The resistor requires a 15V forming pulse to make the device operational. On a standard CMOS processes, 15V is bigger than reverse breakdown voltages for the diodes which are integral to a CMOS transistor.

2 The voltage across the device must be restricted to less than 1V otherwise it will reprogram itself.

A simple 5 by 4 array of the programmable resistors with operational amplifier circuits to convert the summed currents into voltages has successfully solved the benchmark XOR problem. Other proposals include using the resistors linked to a current source to set the synaptic weight voltages.

Both AT&T Bell Laboratories [95] and the Jet Propulsion Laboratory (JPL) [96] have implemented large scale binary matrices with amorphous silicon devices. The Bell device required the use of electron-beam lithography to define which resistors are connected in the matrix array. Thus the function of the device was fixed by the final processing steps. The JPL chip was like an EPROM in that it was programmable electrically once only.

Advantages	Disadvantages
Small Area High Speed/Area Product Continuous Representation (No Quantisation)	Process Variant Accuracy Limited by Noise

Table 3.2 The Advantages and Disadvantages of Analogue Multipliers.

3.3.7. Conclusions

As the demerits and merits listed in Table 3.2 show, the main advantage of analogue multipliers is the small silicon area which they occupy. The high level of synaptic integration possible means that a complete network containing 1000's of synapses can be built on a single integrated circuit. The much larger size of digital multipliers would allow only a small fraction of such a network to be put on an integrated circuit of similar size. Thus analogue circuits offer the potential for completely parallel, and asynchronous, neural implementations in silicon.

Technique	Small	Non-Volatile	Programmable	Speed of Programming
Capacitor	Yes	No	Yes	Fast
MNOS/FGMOS Transistor	Yes	Yes	Yes	Slow
CCDs	Yes	No	Yes	Fast
Amorphous Silicon	Yes	Yes	Yes	Fast

Table 3.3 The Advantages and Disadvantages of Analogue Weight Storage Techniques.

Process variations will cause the synapse response within an analogue synaptic array to be non-uniform, but both this and the effects of noise can be controlled by careful design.

Table 3.3 illustrates the pertinent features of the weight storage techniques outlined in Section 3.3.6. While amorphous silicon appears to have all the desired features for weight storage, the technology is still in the early stages of development. The slow programming speed of MNOS/FGMOS transistors greatly limits the speed of the system when the weights are being constantly modified for example during the training phase of

a Kohonen or a MLP network. Of the two remaining techniques, capacitive storage is compatible with a wide range of analogue multipliers, due to the storage of the weight as a voltage. Since CCD's deal in packets of charge a different type of multiplier needs to been designed [92]. CCD's also require a specialist fabrication process.

3.4. On-Chip Learning

The complexity of the back-propagation learning for MLP networks means that in most analogue VLSI MLP implementations the modifications to the synaptic weights are normally calculated by the host computer. Variants of the simpler Hebbian learning have however been implemented in analogue VLSI [97, 98, 63]. As learning will not be included in the proposed implementation, further discussion of this topic is not appropriate for this review chapter. Descriptions of the Hebbian learning implementations mentioned earlier are however contained in Appendix 3.

3.5. Pulse Based Implementations

In biological neurons the neuron state is encoded as the firing rate of a train of pulses. This inspired Murray [99, 100] to propose VLSI neural networks where the neural information is represented as a pulse stream.

Figure 3.21 shows 4 ways of coding neural state information, V_i, using a pulse stream.

1 Pulse Amplitude Modulation (PAM) : the height of the fixed-frequency pulses varies with V_i.

2 Pulse Width Modulation (PWM) : the frequency of the pulses is fixed but the width of pulse varies with V_i.

3 Pulse Frequency Modulation (PFM) : the frequency of the fixed-width pulses is directly proportional to V_i.

4 Pulse Density Modulation (PDM) : the number of fixed-width pulses per second is directly proportional to V_i. This differs from PFM in that the inter-pulse spacing is not necessarily related to V_i.

5 Pulse Phase Modulation (PPM) : V_i is encoded as the phase difference between the pulses in 2 separate pulse trains.

To help differentiate between voltage and pulse representations of neural states, in the remainder of this thesis, a neural state encoded as a pulse will be referred to as S_i with V_i signifying neural states represented as voltages.

The latter 4 techniques have the advantage that the magnitude of the pulse does not vary and thus can assume digital levels. PPM requires an extra line to encode the state

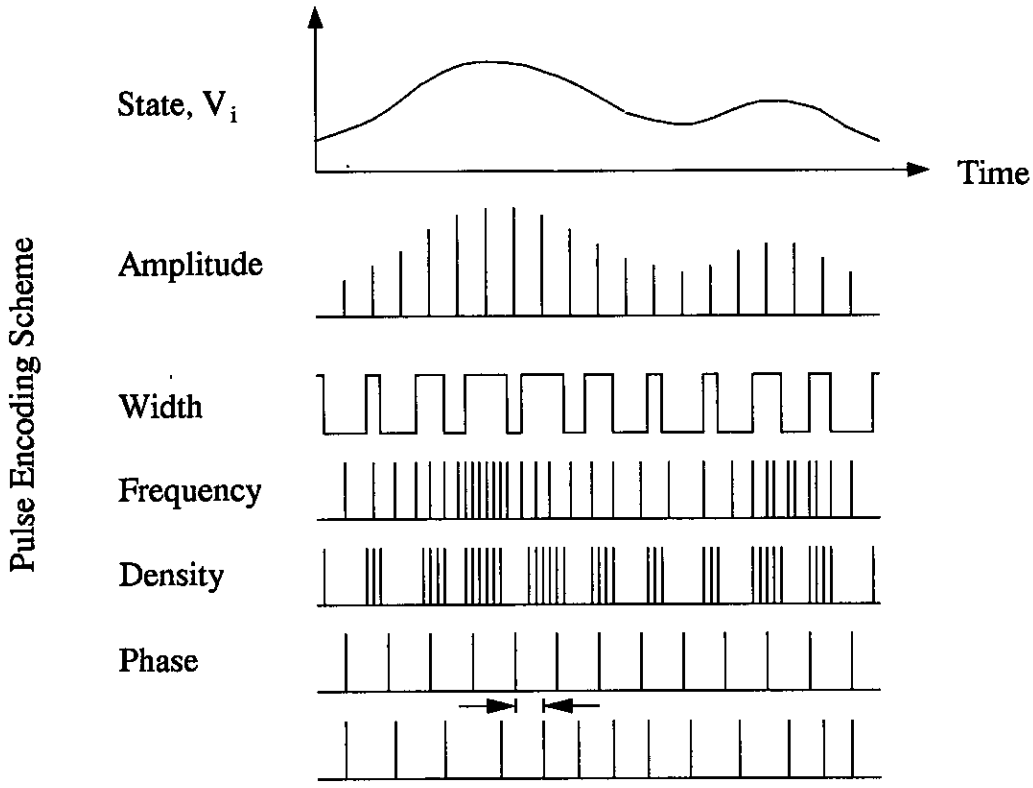


Figure 3.21 Pulse Stream Encryption.

data. For this reason, PWM, PFM and PDM are the most commonly used of these techniques. In all of these cases the neural state information is encoded in the time domain.

Circuits based on pulse encoding have the robust nature of digital signal levels and the compactness of an analogue multiplier. For example in a digital implementation the multiplication of 2 *stochastic* bit streams is achieved via an AND gate, whereas in an analogue implementation the multiplier reduces to a variable magnitude current source the output of which is gated by the pulse stream. The distribution of the multiplication in time inevitably slows the computation rate of a pulse based system. However, while individual biological neurons have very slow response times compared with the speed of present day computers, the human neural system is able to outperform the same computers at image processing tasks due to its massively parallel nature. Thus the speed limitations of pulse based systems should not be viewed as a major problem, especially when the compactness of its circuits will allow higher levels of synaptic integration in silicon.

Section 4.1 describes the history of pulse based circuits within the author's research group.

Chapter 4

The Design of a Process Invariant Neural Network

The comparison of the Hopfield/Tank and the Kohonen neural networks for the 10 city TSP in Table 4.1 shows clearly that the Hopfield/Tank network requires by far the greater number of synapses and neurons. Thus the Hopfield/Tank network fixes the minimum size of the VLSI implementation at 100 neurons and 10000 synapses.

The requirement for a large scale VLSI network, coupled with the desire for a completely parallel implementation of the synaptic array, dictates the use of analogue VLSI design techniques. To increase further the levels of synaptic integration and to retain some of the benefits of digital signal levels, pulse stream encoding was used.

Network	Cities	Neurons	Synapses
Hopfield/Tank	N	N^2	N^4
	10	100	10000
Kohonen	N	2N – 3N	6N – 9N
	10	20-30	60-90

Table 4.1 Network Specifications of the TSP.

As was discussed in Section 3.1 the limitation of analogue circuits is their sensitivity to the tolerances of the fabrication process. An important consequence of such variations is that the characteristics of analogue circuits such as multipliers will vary across and between chips. In software simulations of neural networks the characteristics of the synapses and the neurons are completely uniform. Thus as the the hardware is behaving differently from the simulation model, the performance of a neural network implemented in analogue hardware is degraded.

One solution to this problem is to use a network and an algorithm, eg an MLP network and the Back-Propagation learning rule, which compensate for process variations, at least to a first order. Provided the analogue VLSI chip is included in the learning cycle, the weight-set will evolve in a way which compensates for synapse mis-matching.

Such a solution is not suitable for Hopfield/Tank and Kohonen neural networks. These neural networks rely on accurate multiply and accumulate operations for the

algorithms to be effective. It is important to note, that the main aim of this thesis was to maximise the accuracy of neural VLSI implementations and not to carry out an investigation into the accuracy required by the Hopfield/Tank and Kohonen neural networks to successfully solve the TSP.

An alternative approach, is to develop circuit techniques which minimise the harmful effects of process variations. If the circuits can be made sufficiently accurate, then the degradation of the networks performance can be greatly reduced. This approach avoids the need to develop new neural optimisation algorithms which compensate for process variations, and which in any case would probably be both problem- and network-specific. Furthermore, it yields a solution which will extend the range of applications for analogue VLSI in general and for neural networks in particular.

The next section reviews pulse based implementations within and outside this research group to put the circuits developed later in the chapter into context.

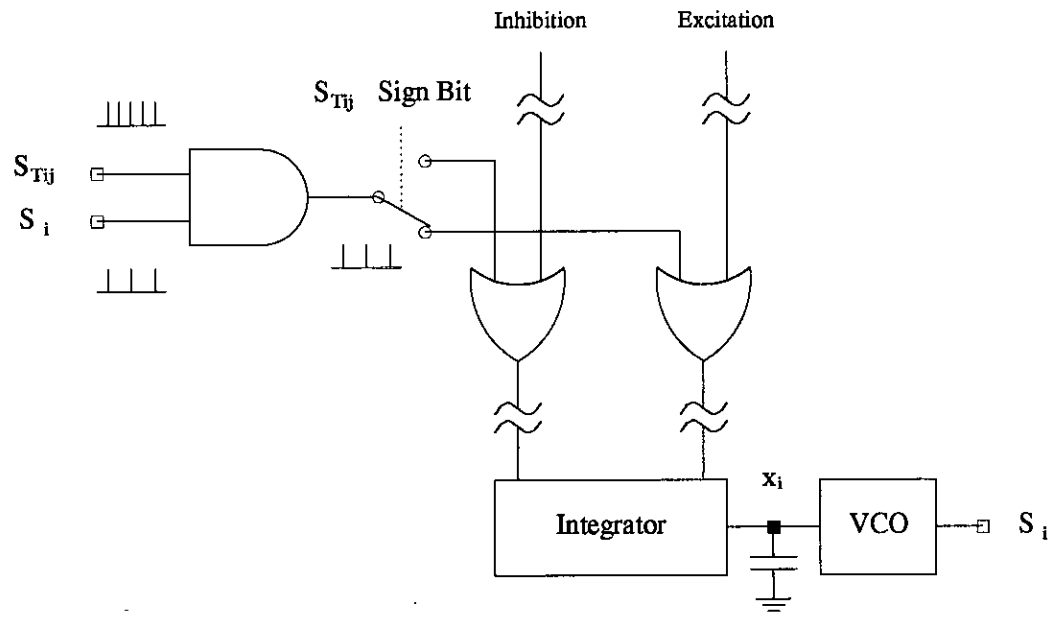


Figure 4.1 Multiplication and Addition of Stochastic Pulse Streams.

4.1. An Overview of Pulse Based Implementations

4.1.1. Digital Implementations

In digital pulse based implementations, if the synaptic weight is also encoded as a PFM pulse train, S_{Tij}, the synapse multiplier simplifies to a single AND gate (Figure 4.1). This can be proven by assigning to each pulse stream a probability equal to the value the

pulse train is representing, ie P(A)=S_{Tij} and P(B)=S_j. If P(A) and P(B) are statistically independent then

$$P(A \text{ AND } B) = P(A).P(B) = S_{Tij}S_j \qquad (4.1)$$

These signals are easily summed via an OR gate. To achieve positive and negative weights, two summation lines feed into an integrator which calculates the difference between the signals. A Voltage Controlled Oscillator (VCO) can be used to convert this voltage into a pulse stream. The main problem with using stochastic pulse trains in such a way, is that long integration times are required to obtain an accurate answer.

Murray *et al* [100] Neural Semiconductor [101] Ricoh [102] and Spaanenburg *et al* [103] have all developed digital VLSI implementations based on PFM/PDM pulse trains. The Neural Semiconductor work is interesting as the implementation is a commercially available product. Each chip contains a 32 by 32 synaptic array and 32 neurons on a standard CMOS process. The chips are also cascadable.

4.1.2. Switched-Capacitor Implementations

In switched-capacitor circuits a resistance is implemented by clocking charge on and off a capacitor. If a capacitor, C, is clocked at a frequency, f, then its equivalent resistance is 1/fC. As this frequency is effectively a PFM pulse train, standard switched-capacitor circuits can be used to implement neural networks. This idea was first proposed by Tsividis in 1987 [104].

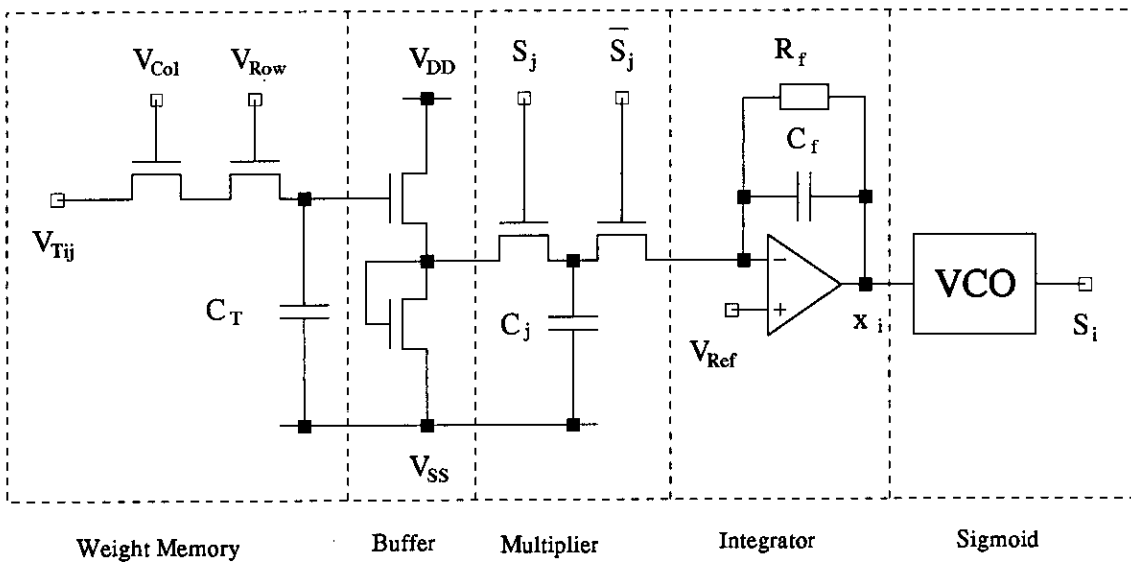


Figure 4.2 Switched-Capacitor Pulse Stream Synapse and Neuron.

The work by Brownlow *et al* at Oxford [105-107] is a good example of the technique. The weight voltage is buffered in the synapse by a simple source follower buffer (Figure 4.2). The value of this voltage determines how much charge is stored on capacitor, C_j. The pulse frequency of the complementary signals S_j and $\bar{S}_j$, fixes how often these packets of charge are transferred to the integrator. Thus the total charge transferred over a period of time is proportional to the weight voltage multiplied by the pulse frequency.

The synapse, including the weight address circuitry and the buffer stage, only occupies 65μm by 65μm, emphasising the compact nature of switched-capacitor circuits. An additional advantage of this circuit is that as the transfer function depends on the ratio of capacitors, C_j/C_f, the transfer function remains very stable as the fabrication process varies, giving accurate multiplications. The software/hardware comparison reported, found the synapse to be accurate within 1.2% [106].

There are however, scalability and noise problems with this circuit. Every time the network is scaled, the current and capacitive load on the operational amplifier increases; thus each time the array size increases the operational amplifier needs to be re-designed. The problem of noise from the large number of pulse streams which exist in a reasonably sized network afflicts all pulse stream implementations and not just this particular synapse.

The Tomberg switched-capacitor implementation [108, 109] is similar in concept to Brownlow's implementation but the weight is stored as a 16-bit PDM pulse stream with the neural state also represented by a PDM signal. Due to digital weight storage the synapse area ($50,000\mu m^2$) is 10 times that of the Brownlow synapse.

4.1.3. Analogue Implementations

Using pulse streams to encode neural states simplifies synapse circuitry. A synapse becomes a voltage controlled current source with a switch to gate its output. The frequency or width of the output current pulses is modulated by the neural state pulse stream while the amplitude is controlled by the synaptic weight voltage via the voltage controlled current source. The area under these current pulses is thus proportional to product of the neural state and the synaptic weight.

In PWM implementations the synaptic multiplications are usually synchronised [110, 111] while in PFM circuits the pulse trains are generated by free running oscillators [112] so the update is asynchronous. As the simulation results in Chapter 2 show, the order of updating neural states in the fully connected neural networks influences the choice of minimum "found" by the network.

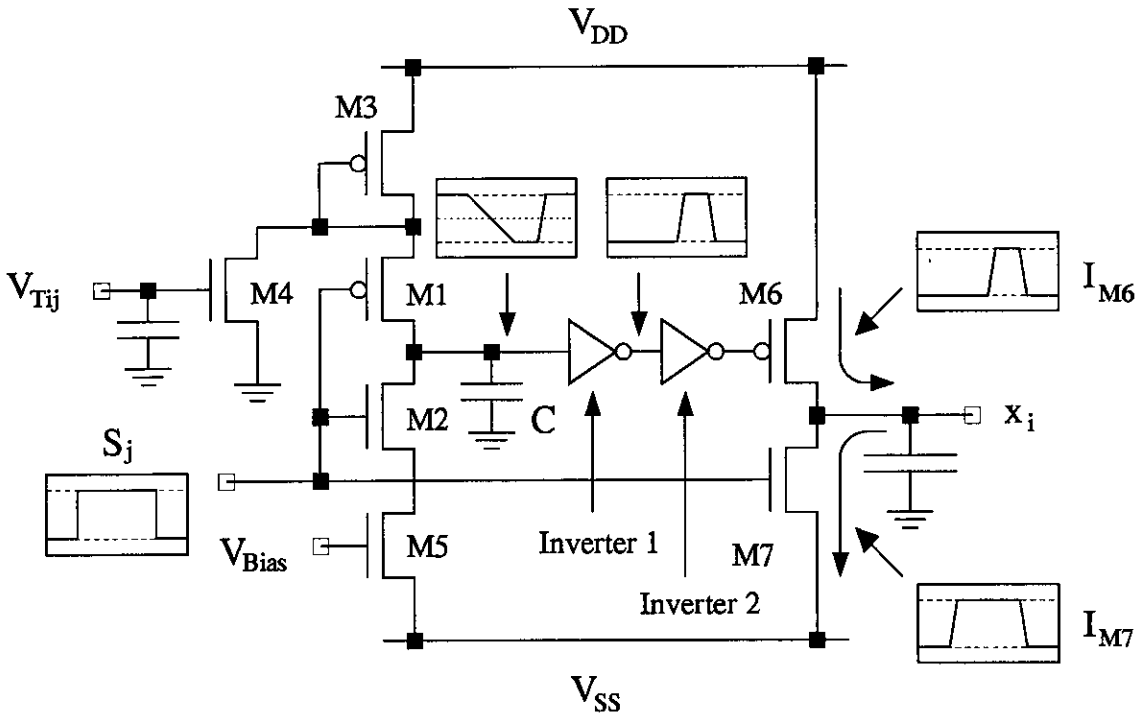


Figure 4.3 Programmable Pulse Width Modulation Synapse.

Such hybrid implementations can have many advantages. The digital states are robust and easily regenerated while the analogue multipliers are compact and provide a continuous representation for the synaptic weight.

An early PFM synapse (Figure 4.3) by Hamilton [113-115, 112] used fixed value current sources (as opposed to a variable current sources). Multiplication is obtained by "chopping" the pulse width of the incoming pulse stream. Varying the neural state changes the frequency of the fixed amplitude current pulses output from the synapses while decreasing the synaptic weight decreases the width of these pulses.

The modulation of the incoming pulse width is achieved as follows. When a pulse is received capacitor C starts discharging. The initial voltage for this discharge is determined by the synaptic weight. A higher initial voltage will result in a longer time for the constant rate discharge to reach the switch point of inverter 1. At the switch point, the output of inverter 1 goes high. When the input pulse ends the voltage on the capacitor returns very quickly to its preset voltage, taking inverter 1's output low. Thus there is a linear relationship between the initial discharge voltage and the pulse width generated by inverter 1. This variable pulse width controls the size of the charge dumped on to the activity capacitor by transistor M6. At the same time transistor M7, under the control of the input pulse, removes enough charge so that when the weight is at its zero voltage, the net charge added to the activity capacitor is zero.

The synapse in the fabricated 10 by 10 test array occupied 173μm by 73μm. The results reported confirm the successful operation of the synapse. Unfortunately the test results also show that the synapse is sensitive to the effects of process variation.

4.2. A Process Invariant Synapse

4.2.1. The Transconductance Multiplier

At the Neural Information Processing Systems conference in 1989 [115], the author's research group proposed 2 pulse stream (PFM) synapses based on linear transconductance multipliers [72, 73] (Figure 4.4). The attractions of these two circuits are two-fold.

1 Compactness : only 3 or 4 transistors are required.

2 A highly linear input/output characteristic.

The principles behind the operation of the 4 and 3 transistor variants are similar. In both cases transistors M1 and M2 form a transconductance multiplier. The output current from these transistors, I_{ij}, which is proportional to the synaptic weight voltage V_{Tij}, is pulsed at a frequency controlled by the incoming neural state S_j. In the 3 transistor synapse this pulsing action is achieved via switching transistor M3. However, in the 4 transistor variant, the gating of current, I_{ij}, is accomplished using transistors M3 and M4 to switch the whole of the transconductance multiplier in and out. By integrating the resultant output current over a period of time the required $V_{Tij}S_j$ multiplication is achieved. A Voltage Controlled Oscillator (VCO) then converts this activation level into a stream of fixed width pulses.

The operation of these two synapses can be explained with reference to the I_{DS} equation for a MOSFET transistor in its linear region of operation. Equation 4.2 shows that for a such a transistor the drain-source current, I_{DS}, is proportional to the gate-source voltage, V_{GS}, multiplied by the drain-source voltage, V_{DS}.

$$I_{DS} = \frac{\mu_O \varepsilon_{OX}}{t_{OX}} \frac{W}{L} \left[V_{GS}V_{DS} - V_T V_{DS} - \frac{V_{DS}^2}{2} \right] \tag{4.2}$$

where

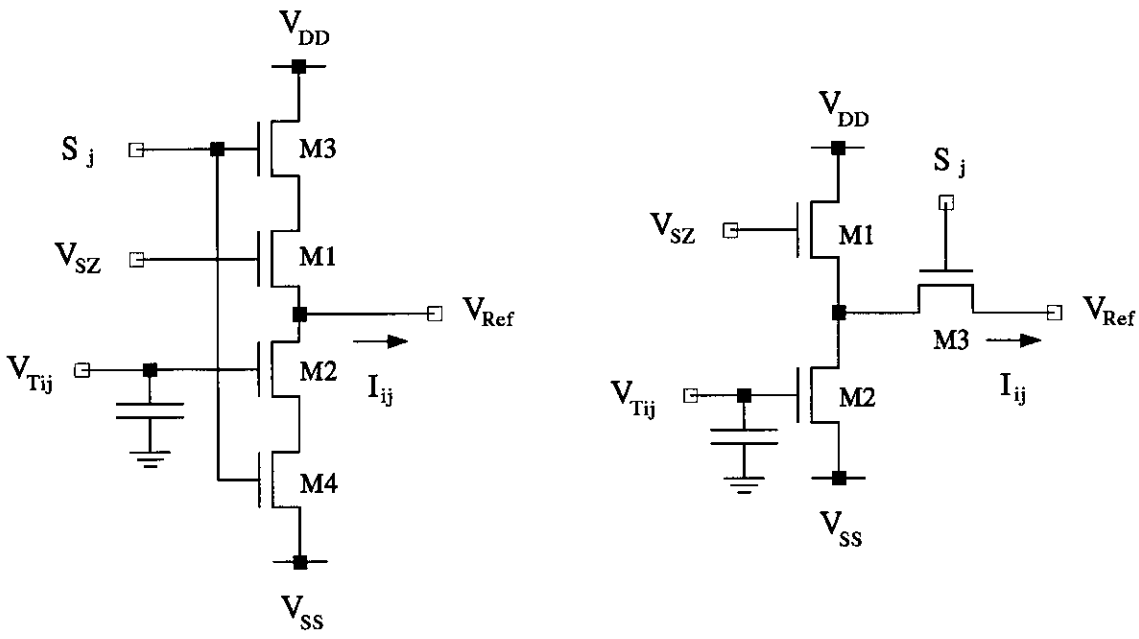
$(V_{GS} - V_T) > V_{DS}$

μ_O = Surface mobility of the channel ($cm^2/V/s$)

ε_{OX} = Permittivity of Silicon Dioxide (3.45×10^{-13}F/cm)

t_{OX} = Thickness of the transistor's gate oxide (m)

W = Effective transistor gate width (m)



(a) 4 Transistor Synapse. (b) 3 Transistor Synapse.

Figure 4.4 Transconductance Multiplier Synapses.

L	=	Effective transistor gate length (m)
V_T	=	Transistor's threshold voltage (V)

Unfortunately in Equation 4.2, two non-linear terms are present in addition to the desired $V_{GS}V_{DS}$ term. A second transistor, M1, which is identical to M2, is used to eliminate these terms. For the second and third terms of Equation 4.2 to be cancelled exactly the V_{DS} and V_T must also be the same for both transistors. It is straightforward to arrange for V_{DS1} to equal V_{DS2}. However, transistors M1 and M2 have different V_{BS}, and thus have different threshold voltages (due to the body effect). To investigate whether this difference, $(V_{T1} - V_{T2})$, is small enough to neglect, it is necessary to calculate the shifts in threshold voltage caused by the body effect.

The threshold voltages for the NMOS transistors M1 and M2 are given by the equations

$$V_{T1} = V_{T0} + \gamma\left[\sqrt{2|\phi_F| + V_{BS1}} - \sqrt{2|\phi_F|}\right] \tag{4.3}$$

$$V_{T2} = V_{T0} + \gamma\left[\sqrt{2|\phi_F| + V_{BS2}} - \sqrt{2|\phi_F|}\right] \tag{4.4}$$

where

γ = Bulk Threshold Parameter ($V^{\frac{1}{2}}$) = $\frac{t_{OX}}{\varepsilon_{OX}}\sqrt{2\varepsilon_{Si}qN_{SUB}}$

ϕ_F = Strong Inversion Surface Potential (V) = $\frac{kT}{q}\ln\left(\frac{N_{SUB}}{n_i}\right)$

V_{T0} = Threshold Voltage for $V_{BS} = 0$ (V)

N_{SUB} = Substrate Doping Concentration (cm^{-3})

n_i = Intrinsic Carrier Concentration ($1.45 \times 10^{10} cm^{-3}$)

ε_{Si} = Permittivity of silicon ($1.0359 \times 10^{-12} F/cm$)

k = Boltzmann's Constant ($1.381 \times 10^{-23} J/^{\circ}K$)

T = Temperature (300°K)

q = Charge on an Electron ($1.6 \times 10^{-19} C$)

Parameter	ES2 2μm Process		
	L2 - Fast	L2 - Typical	L2 - Slow
V_T	0.7V	0.9V	1.0V
t_{OX}	38nm	40nm	42nm
μ_O	$530 cm^2/V/s$	$510 cm^2/V/s$	$480 cm^2/V/s$
N_{SUB}	$0.48 \times 10^{16} cm^{-3}$	$0.53 \times 10^{16} cm^{-3}$	$0.60 \times 10^{16} cm^{-3}$

Table 4.2 The Variation of Selected N-Type Transistor Parameters.

Thus the difference in threshold voltages is

$$\Delta V_T = V_{T1} - V_{T2} = \gamma\left[\sqrt{2|\phi_F| + V_{BS1}} - \sqrt{2|\phi_F| + V_{BS2}}\right] \tag{4.5}$$

If $V_{SS} = 0$, $V_{DS1} = V_{DS2} = 0.8V$ and $V_{DD} = 1.6V$ then $V_{BS1} = 0.8V$ and $V_{BS2} = 0V$. Using the process parameters given in Table 4.2 gives ΔV_T as approximately $0.2V \pm 10\%$. This value is significant when compared to $(V_{GS1} - V_{GS2})$ and thus in this case it is not valid to assume that $\Delta V_T = 0$.

Therefore the output current of the transconductance synapse is given by

$$I_{ij} = \frac{\mu_O \varepsilon_{OX}}{t_{OX}} \frac{W}{L}\left[((V_{GS1} - \Delta V_T) - V_{GS2})V_{DS}\right] \tag{4.6}$$

Thus the current, I_{ij} is directly proportional to the voltage difference $((V_{GS1} - \Delta V_T) - V_{GS2})$ multiplied by V_{DS}.

In this implementation the transconductance multiplier is being used as a voltage controlled current source. This is achieved by keeping the V_{DS} voltages constant and only varying V_{GS1}. Thus the output current is now proportional to V_{GS1} which represents

the synaptic weight. V_{GS2} determines the value of V_{GS1} at which the output current is zero. The output of the circuit M1/M2/M3 is therefore a stream of pulses whose magnitude is proportional to V_{Tij}, with a frequency S_j pulses/second and a pulse width Δt. Alternatively, the area under the pulse stream can be expressed as a duty cycle, DC_j where $DC_j=\Delta t S_j$. Thus the charge being transferred onto the integration capacitor over the time interval t_1 to t_2 by synapse ij is

$$Q_{ij} = (t_2 - t_1) \times DC_j \times \frac{\mu_O \varepsilon_{OX}}{t_{OX}} \frac{W}{L} \left[((V_{GS1} - \Delta V_T) - V_{GS2}) V_{DS} \right] \tag{4.7}$$

Extending this to a column of N synapses gives the following equation for the charge being sourced/sunk from the integration capacitor in over the time interval t_1 to t_2.

$$Q_i = (t_2 - t_1) \times \frac{\mu_O \varepsilon_{OX}}{t_{OX}} \frac{W}{L} V_{DS} \sum_{j=0}^{N-1} \left[DC_j \times ((V_{GS1} - \Delta V_T) - V_{GS2}) \right] \tag{4.8}$$

This confirms that the charge being transferred is indeed proportional to the sum of the synaptic weights, V_{GS2}, multiplied by their appropriate pulse frequences, S_j.

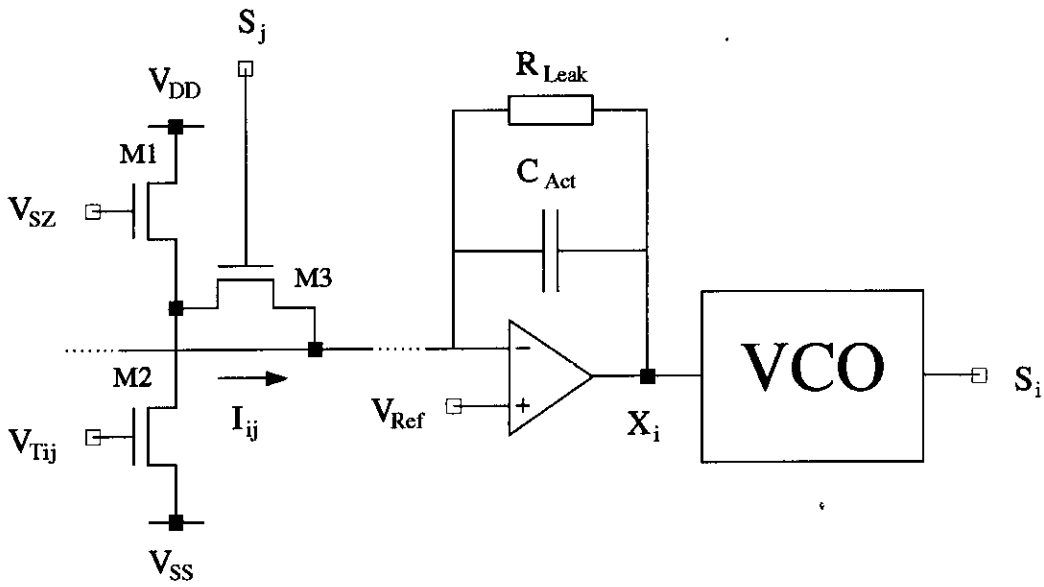


Figure 4.5 The 3 Transistor Synapse and Neuron.

As Figure 4.5 shows, the neuron for the transconductance synapse is composed of a "leaky" integrator and a Voltage Controlled Oscillator (VCO). The "leaky" integrator sums the packets of charge from a column of these transconductance synapses, converting them into the neuron's activity voltage. This voltage then controls the duty cycle of the VCO which has a sigmoidal transfer characteristic.

Thanks to the small numbers of transistors, and to the fact that all the transistors are N-types (there are no area-hungry well crossings), this synapse occupies an area of only 100μm by 100μm. This is one of the main advantages of the synapse, allowing 1000's of synapses to be implemented on a single integrated circuit.

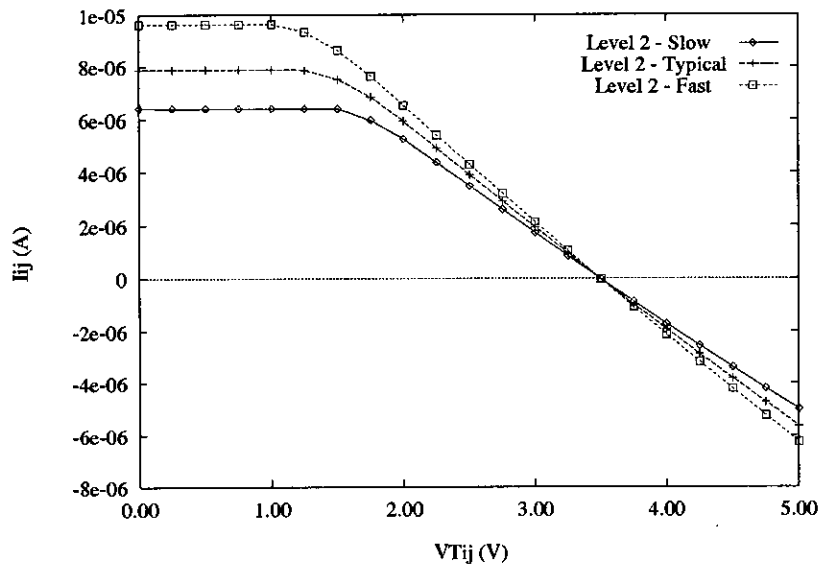


Figure 4.6 Simulated Process Variation for the Transconductance Multiplier.

HSPICE simulations using the HSPICE Level 2 transistor models for the European Silicon Structures 2μm process, revealed four problems with this system.

1 Due to the large settling time, incurred by switching the whole of the transconductance stage and its associated capacitance in and out, the 4 transistor synapse had a poor transient response. As the 3 transistor synapse pulses only the output current from the transconductance stage, it has a superior transient response.

2 The results in Table 4.3 and Figure 4.6 show the transconductance synapse's voltage to current relationship for the "fast", "typical" and "slow" process parameters. While the voltage to current relationship is very linear, the **slope** of the response varies by approximately ±11%. Equation 4.6 shows that variations in the surface mobility of the transistor channel, μ_O, and the gate oxide thickness, t_{OX}, have a direct effect on the magnitude of the output current. The variation in ΔV_T is ±20mV (±10% of 0.2V) and thus can be ignored. The process parameters in Table 4.2, reveal that the tolerance on t_{OX} is ±5% while for μ_O it is -6%/+4%. This accounts for the overall 11% variation.

V_{Tij} (V)	I_{ij} (μA)			Percentage Variation
	L2 - Fast	L2 - Typical	L2 - Slow	
1.00	6.427	7.885	9.646	+25.9%/-31.3%
2.00	5.277	5.948	6.534	+11.9%/-10.4%
3.00	1.722	1.936	2.132	+3.8%/-3.5%
4.00	-1.701	-1.917	-2.120	+3.6%/-3.8%
5.00	-4.990	-5.634	-6.241	+10.8%/-11.4%

Table 4.3 Percentage Process Variation for the Transconductance Synapse.

3 The output current is sensitive to the value of the mid-point voltage between transistors M1 and M2. To maintain this point to 1% of the transistors V_{DS}, the integrator must include a 2 stage operational amplifier. Other amplifiers (such as simple inverters) have neither the drive capability nor a high enough gain (>1000).

4 As the number of synapses per neuron is scaled up, both the maximum current and the capacitance will also increase. The operational amplifier in the "leaky" integrator has a finite current drive capability and is only compensated up to a specified capacitive load, so every time the system is scaled the operational amplifier must be redesigned. This effectively renders the system uncascadable.

The combination of limited cascadability and process variance renders this system unsuitable for implementing neural networks in the form just described.

4.2.2. A Synapse Based on Distributed Feedback

To solve the problems of process variation and poor cascadability, a buffer stage (M4 and M5 in Figure 4.7) was added to the transconductance synapse. As both of these transistors operate in their linear regions, this buffer stage is effectively another transconductance multiplier. The operational amplifier at the foot of each post-synaptic column provides a feedback signal, V_{outi}, that controls the current in all the buffer stages in that column of synapses such that the buffer current balances the current being sourced or sunk by the multipliers. The gate voltage of transistor M5, V_{Bias}, determines the voltage level about which V_{outi} varies. Thus the buffer stage plus the feedback operational amplifier is functionally equivalent to a standard operational amplifier current to voltage converter, where the resistor in the feedback loop has been replaced by transistor M4 (Figure 4.8). As in Section 4.2.1, the current pulses from the transconductance stage are summed within a synaptic column, at the common output node of transistor M3. The buffer stage under the control of the feedback operational amplifier converts the stream of the

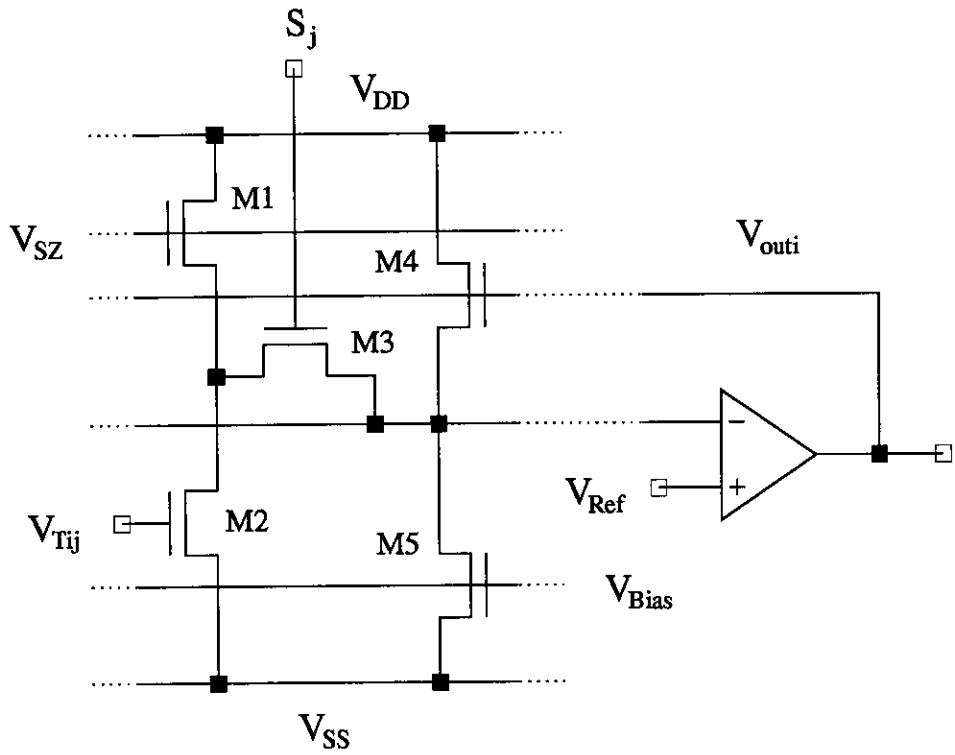


Figure 4.7 A Synapse Based on Operational Amplifier Feedback.

summed current pulses into a stream of voltage pulses. The process invariant voltage integrator, discussed in Section 4.3.2, then integrates the area under the stream of voltage pulses, to yield the required multiplication result.

The problem of cascadability has now been eased, as the operational amplifier drives only transistor gates, instead of a large integration capacitor. Also, the current demanded by the transconductance synapses is now supplied by the distributed buffer stage rather than by the operational amplifier. As the current and capacitive drive demands are now less, the resulting operational amplifier is much more compact.

Assuming that the buffer and transconductance stages are well matched, then their voltage to current characteristics will be very similar. Thus the overall voltage to voltage transfer function for the synapse should remain constant even although the voltage to current functions for the buffer and transconductance stages vary with process imperfections. To confirm this process invariance, a more formal examination of the synapse is required. The analysis of a column of N synapses is simplified by breaking the synapse into its two components: the transconductance synapse and the buffer stage. By assuming that when M3 is ON the voltage across it is zero, the synapse can be analysed as a pair of back to back transconductance multipliers (Equation 4.6), where V_{SS} = 0V and $V_{DD} = 2V_{Ref}$. Thus the current output by the transconductance stages within the synaptic column is

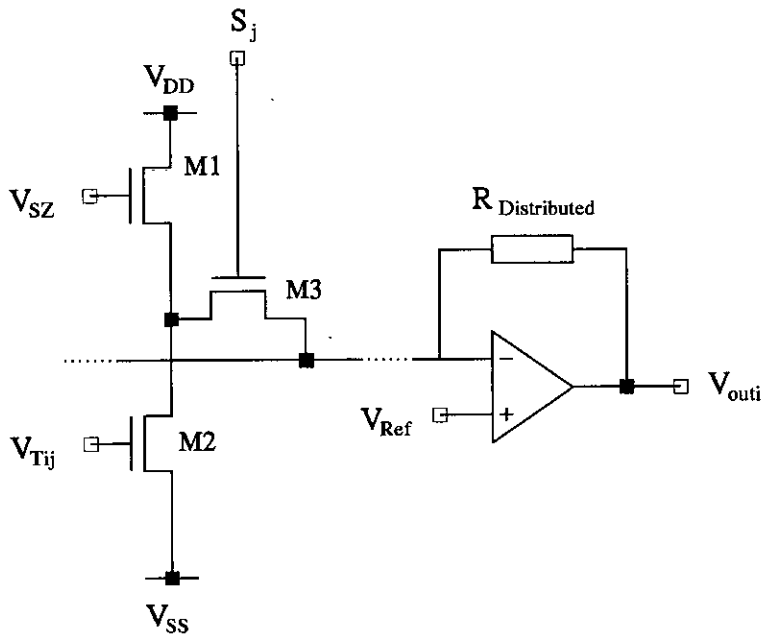


Figure 4.8 The Equivalent Circuit for the Distributed Feedback Synapse.

$$I_{Trans} = \beta_{Trans} \sum_{j=0}^{N-1} ((V_{SZ} - V_{Ref} - \Delta V_T) - V_{Tij}) V_{Ref} \tag{4.9}$$

where $\beta_1 = \beta_2 = \beta_{Trans} = \dfrac{\mu_O \varepsilon_{OX}}{t_{OX}} \dfrac{W_{Trans}}{L_{Trans}}$.

Similarly for the N buffer stages

$$I_{Buf} = N\beta_{Buf}((V_{outi} - V_{Ref} - \Delta V_T) - V_{Bias}) V_{Ref} \tag{4.10}$$

where $\beta_4 = \beta_5 = \beta_{Buf} = \dfrac{\mu_O \varepsilon_{OX}}{t_{OX}} \dfrac{W_{Buf}}{L_{Buf}}$.

The feedback operational amplifier ensures that the current in the buffer stages balances the current in the transconductance stages i.e.

$$I_{Trans} + I_{Buf} = 0 \tag{4.11}$$

Substituting for I_{Trans} and I_{Buf} in Equation 4.11 gives

$$\beta_{Trans} \sum_{j=0}^{N-1} ((V_{SZ} - V_{Ref} - \Delta V_T) - V_{Tij}) V_{Ref} = \tag{4.12}$$

$$-N\beta_{Buf}((V_{outi} - V_{Ref} - \Delta V_T) - V_{Bias}) V_{Ref}$$

Solving for V_{outi} yields

$$V_{outi} = \frac{1}{N}\frac{\beta_{Trans}}{\beta_{Buf}}\sum_{j=0}^{N-1}(V_{Tij} - (V_{SZ} - V_{Ref} - \Delta V_T)) \qquad (4.13)$$

$$+ V_{Bias} + V_{Ref} + \Delta V_T$$

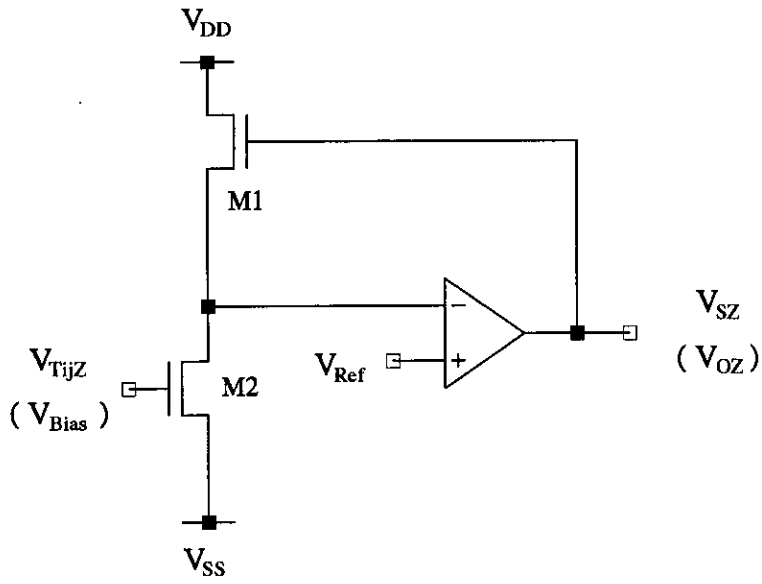


Figure 4.9 The use of Feedback to determine V_{SZ} and V_{OZ}.

On-chip feedback loops incorporating a transconductance multiplier and an operational amplifier determine the values of V_{SZ} and V_{OZ} automatically from the input voltage references V_{TijZ} and V_{Bias} respectively (Figure 4.9). The V_{SZ} feedback loop balances the transconductance stages so that there is no output current when V_{Tij}=V_{TijZ} (zero weight). The second feedback loop calculates the "zero" voltage around which the output of the feedback operational amplifier varies, thus generating the correct balance voltage for the voltage integrator. These mechanisms compensate for process variations between chips. By analysing these circuits in a similar manner to those of the transconductance synapse the following equations are derived:

$$V_{SZ} = V_{TijZ} + V_{Ref} + \Delta V_T \qquad (4.14)$$

$$V_{OZ} = V_{Bias} + V_{Ref} + \Delta V_T \qquad (4.15)$$

Substituting into Equation 4.13 then gives

$$V_{outi} - V_{OZ} = \frac{1}{N}\frac{\beta_{Trans}}{\beta_{Buf}}\sum_{j=0}^{N-1}(V_{Tij} - V_{TijZ}) \qquad (4.16)$$

This is the voltage to voltage characteristic for the synaptic column. Now to find the area

under the output voltage pulse train, let the 0-5V input pulse streams to the synaptic column have duty cycles, DC_0,.., DC_j,.., DC_{N-1}. This gives the area over the time interval t_1 to t_2 as

$$\int_{t_1}^{t_2}(V_{outi} - V_{OZ})dt = (t_2 - t_1) \times \frac{1}{N} \frac{\beta_{Trans}}{\beta_{Buf}} \sum_{j=0}^{N-1}\left[DC_j \times (V_{Tij} - V_{TijZ}) \right] \qquad (4.17)$$

This confirms that the area under the output voltage signal is indeed proportional to the sum of the synaptic multiplications within the column. Furthermore, the output, $(V_{outi} - V_{OZ})$, is dependent on a ratio of β's rather than directly on β, as is the case for the output current of the transconductance multiplier. Since transistors M1, M2, M4 and M5 are close together and are thus well matched, the effects of variations in the surface mobility of the channel and the gate oxide thickness are cancelled to a first order, greatly reducing the effects of process variations on the synapse output voltage.

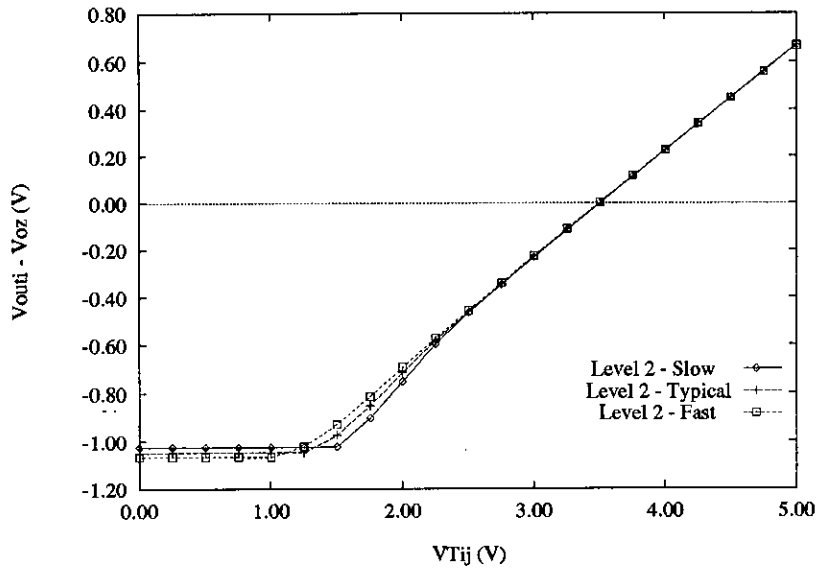


Figure 4.10 Simulated Process Variation for the Synapse.

The results for the simulation of the synapse performance are shown in Figure 4.10. These graphs demonstrate both the linearity of the synapse and its potentially high level of process invariance. The percentage variations for selected results are shown in Table 4.4.

These results indicate that for positive weight values (V_{Tij}>3.5V) the process variation is at least 1 order of magnitude less than that for the same weight voltages applied to the straightforward transconductance multiplier. At low weight voltages, transistor M2

V_{Tij} (V)	$V_{outi} - V_{OZ}$ (V)			% Variation
	L2 - Slow	L2 - Typical	L2 - Fast	
1.00	-1.022	-1.046	-1.064	+3.5%/-2.7%
2.00	-0.751	-0.713	-0.692	+3.2%/-5.7%
3.00	-0.230	-0.228	-0.224	+0.5%/-0.2%
4.00	0.228	0.228	0.229	+0.1%/0.0%
5.00	0.669	0.669	0.668	+0.1%/-0.1%

Table 4.4 Percentage Process Variation for the Distributed Feedback Synapse.

operates on the edge of its linear region. As a result, the matching between it and the other transistors breaks down giving an increase in process variance at the bottom end of the weight range.

This mis-match is also responsible for the decrease in linearity in the synapse characteristic, such that V_{Tij}=2V gives a bigger output swing than V_{Tij}=5V. The condition for the linear operation, $(V_{GS} - V_T) > V_{DS}$, implies that the crossover voltage for linear operation depends directly on the value of the threshold voltage. Thus the degree of this non-linearity is affected by the nature of the process, with a slow process (highest V_T) creating the worst case variations for low weight voltages.

The process within a single chip must remain approximately constant for the synapse to function correctly. This is necessary to ensure that the transconductance stage and the buffer stages are properly matched. However the synapse circuit design and the automatic bias circuitry will compensate for any gross variations between chips.

This compensation feature, combined with the improved cascadability of the system, gives the distributed feedback synapse a clear performance edge over the transconductance multiplier described in Section 4.2.1.

4.3. A Process Invariant Neuron

The matching neuron for the above process-invariant synapse contains 3 components (Figure 4.11).

1 The operational amplifier for controlling the buffer stage of the synapse.

2 The voltage integrator to convert the voltage pulse stream output from the operational amplifier into the aggregated neural activity.

3 A Voltage Controlled Oscillator (VCO) with a sigmoidal transfer characteristic to convert the activity voltage into a stream of variable frequency, fixed width

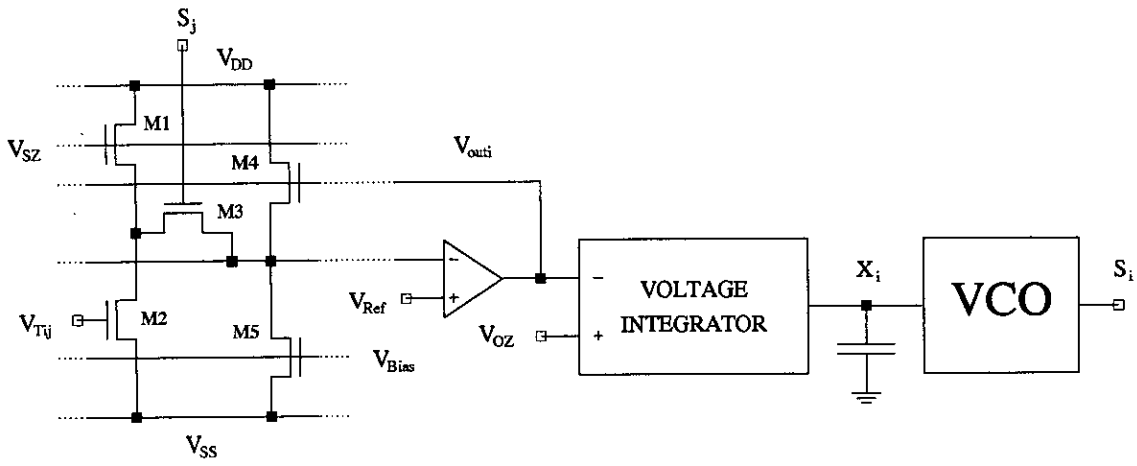


Figure 4.11 An Overview of the Complete System.

pulses.

4.3.1. The Feedback Operational Amplifier

The feedback amplifier must be fast enough to be able to respond to 1μs pulse widths. This gives a slew rate specification of 5V/μs. To maintain V_{Ref} to the required 1% accuracy, the operational amplifier must keep V_{Ref} constant to within 8mV over its 1.6V output range. This gives a minimum amplifier gain of 200 (1.6/0.008). However, in order to provide a good safety margin and to further improve accuracy the operational amplifier was designed with a minimum gain of 1000. The amplifier had also to be able to drive capacitive loads of up to 20pF. To meet these specifications a 2 stage operational amplifier was designed (Figure 4.12).

Before the correct value of the compensation capacitor can be determined the gain around the feedback loop must be known. To calculate the relationship between V_{outi} and V_{Ref} for the distributed feedback synapse, let us first assume that M3 is on and thus V_{DS3} = 0V. Now let

$$V_{SS} = 0V \qquad \beta_1=\beta_2=\beta_{Trans} \qquad \beta_4=\beta_5=\beta_{Buf}$$

Since all of the transistors are in their linear regions of operation the current equations for transistors M1, M2, M4 and M5 are as follows

$$I_{M1} = \beta_{Trans}\left[(V_{SZ} - V_{Ref} - V_{T1})(V_{DD} - V_{Ref}) - \frac{(V_{DD} - V_{Ref})^2}{2}\right] \qquad (4.18)$$

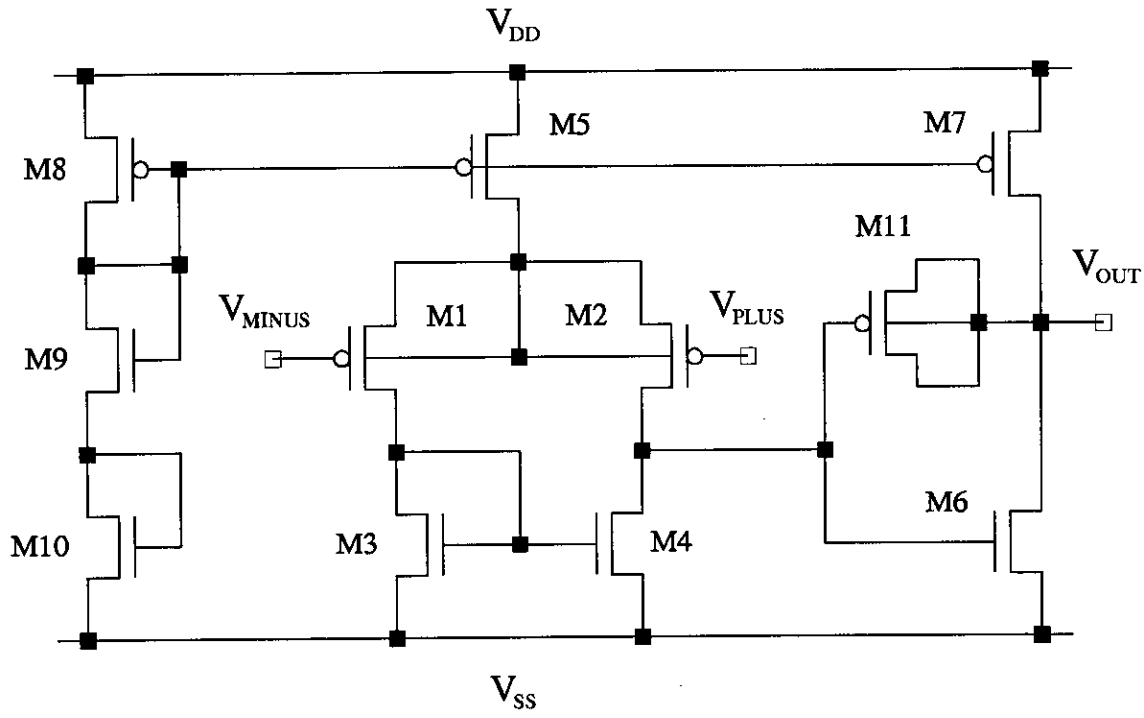


Figure 4.12 A 2 Stage Unbuffered CMOS Operational Amplifier.

$$I_{M2} = \beta_{Trans}\left[(V_{Tij} - V_{T2})V_{Ref} - \frac{V_{Ref}^2}{2}\right] \tag{4.19}$$

$$I_{M4} = \beta_{Buf}\left[(V_{outi} - V_{Ref} - V_{T4})(V_{DD} - V_{Ref}) - \frac{(V_{DD} - V_{Ref})^2}{2}\right] \tag{4.20}$$

$$I_{M5} = \beta_{Buf}\left[(V_{Bias} - V_{T5})V_{Ref} - \frac{V_{Ref}^2}{2}\right] \tag{4.21}$$

Applying Kirchhoff's Current Law yields

$$I_{M1} + I_{M4} = I_{M2} + I_{M5} \tag{4.22}$$

Substituting Equations 4.18 to 4.21 into Equation 4.22 and re-arranging it into a quadratic in terms of V_{Ref} gives

$$(\beta_{Trans} + \beta_{Buf})V_{Ref}^{2} + \tag{4.23}$$

$$\left[\beta_{Trans}(V_{T1} - V_{SZ} + V_{T2} - V_{Tij}) + \beta_{Buf}(V_{T4} - V_{outi} + V_{T5} - V_{Bias})\right]V_{Ref} +$$

$$\left[\beta_{Trans}\left(V_{SZ} - V_{T1} - \frac{V_{DD}}{2}\right) + \beta_{Buf}\left(V_{outi} - V_{T4} - \frac{V_{DD}}{2}\right)\right]V_{DD} = 0$$

As Equation 4.23 has the form

$$aV_{Ref}^{2} + bV_{Ref} + c = 0 \tag{4.24}$$

it can be solved by using the equation shown below

$$V_{Ref} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{4.25}$$

Comparing Equations 4.23 and 4.24 gives the coefficients of the quadratic as follows

$$a = (\beta_{Trans} + \beta_{Buf})$$

$$b = \beta_{Trans}(V_{T1} - V_{SZ} + V_{T2} - V_{Tij}) + \beta_{Buf}(V_{T4} - V_{outi} + V_{T5} - V_{Bias})$$

$$c = \left[\beta_{Trans}\left(V_{SZ} - V_{T1} - \frac{V_{DD}}{2}\right) + \beta_{Buf}\left(V_{outi} - V_{T4} - \frac{V_{DD}}{2}\right)\right]V_{DD}$$

Now to obtain the gain around the feedback loop Equation 4.25 is differentiated with respect to V_{outi} giving

$$\frac{dV_{Ref}}{dV_{outi}} = \frac{1}{2a}\left[\beta_{Buf} \pm \frac{1}{2}\frac{(-2b\beta_{Buf} - 4a\beta_{Buf}V_{DD})}{\sqrt{b^2 - 4ac}}\right] \tag{4.26}$$

The above equation describes the gain around the feedback loop with the transconductance multiplier switched in. To derive the equation for the gain around the feedback loop with the transconductance multiplier switched out, set β_{Trans} to zero in Equation 4.23. As β_{Buf} is now a common factor in Equation 4.23, the equation further simplifies to

$$V_{Ref}^{2} + (V_{T4} - V_{outi} + V_{T5} - V_{Bias})V_{Ref} + \left(V_{outi} - V_{T4} - \frac{V_{DD}}{2}\right)V_{DD} = 0 \tag{4.27}$$

Thus

$$a = 1$$

$$b = (V_{T4} - V_{outi} + V_{T5} - V_{Bias})$$

$$c = \left(V_{outi} - V_{T4} - \frac{V_{DD}}{2}\right)V_{DD}$$

This then gives the gain around the feedback loop with the transconductance stage switched out as

$$\frac{dV_{Ref}}{dV_{outi}} = \frac{1}{2}\left[1 \pm \frac{1}{2}\frac{(-2b - 4V_{DD})}{\sqrt{b^2 - 4c}}\right] \tag{4.28}$$

V_{Tij} (V)	dV_{Ref}/dV_{outi}			
	Theory	HSPICE Level 2		
		Slow	Typ	Fast
2.0	0.258	0.230	0.246	0.221
3.5	0.174	0.190	0.172	0.145
5.0	0.131	0.132	0.122	0.107

Table 4.5a Feedback loop gain with M3 ON.

V_{Bias} (V)	dV_{Ref}/dV_{outi}			
	Theory	HSPICE Level 2		
		Slow	Typ	Fast
2.5	0.308	0.380	0.332	0.269

Table 4.5b Feedback loop gain with M3 OFF.

Tables 4.5a and 4.5b compare the feedback loop gain values calculated using Equations 4.26 and 4.28 with the values obtained from HSPICE Level 2 simulations. The peak in the plot of these results in Figure 4.13 occurs when the transistor M4 crosses into the saturation region. Below are the values of the parameters used in the theoretical gain calculation.

V_{DD}=1.6V V_{Ref}=0.8V V_{SZ}=4.5V V_{Bias}=2.5V
V_{T1}=1.0V V_{T2}=0.8V V_{T4}=1.0V V_{T5}=0.8V

As all of the feedback loop gain values are less than 1, the proposed synaptic feedback loop will remain stable under all weight values.

An interesting feature of this feedback system is that the gain varies with the weight voltage and also according to whether the transconductance stage is switched in or not. This variation results from the variation in the conductance of transistor M4 with its DC bias conditions. An expression for the small signal gain between the gate and source terminals of a transistor can be derived using the conductances g_m and g_{ds}.

$$g_m = \frac{\delta i_D}{\delta v_{GS}} \approx \beta V_{DS} \tag{4.29}$$

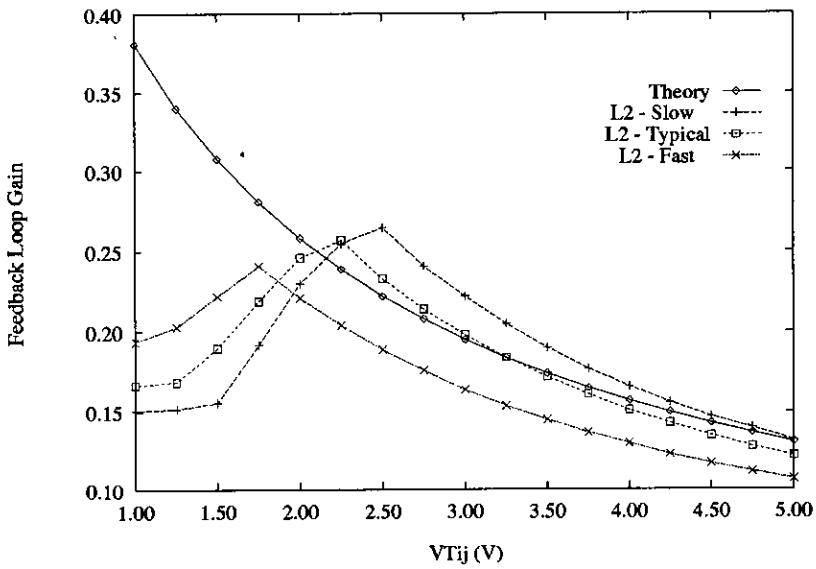


Figure 4.13 Theoretical and Simulated Feedback Loop Gains for the Distributed Feedback Synapse.

$$g_{ds} = \frac{\delta i_D}{\delta v_{DS}} \approx \beta(V_{GS} - V_T - V_{DS}) \tag{4.30}$$

Now noting that the gain can be expressed as a ratio of g_m to g_{ds}.

$$\frac{\delta v_{DS}}{\delta v_{GS}} = \frac{\delta v_{DS}}{\delta i_D}\frac{\delta i_D}{\delta v_{GS}} = \frac{g_m}{g_{ds}} \tag{4.31}$$

The desired small signal gain is then given by

$$\frac{\delta v_{DS}}{\delta v_{GS}} = \frac{\beta V_{DS}}{\beta(V_{GS} - V_T - V_{DS})} = \frac{V_{DS}}{(V_{GS} - V_T - V_{DS})} \tag{4.32}$$

Thus as V_{GS} increases the $\delta v_{DS}/\delta v_{GS}$ of a transistor decreases, explaining why the feedback loop gain decreases as the weight voltage increases.

The gain is smaller with the transconductance stage switched in as the larger current flowing at the amplifier's negative terminal means that the current in transistor M4 has a proportionately smaller effect on the voltage value at the amplifier's negative terminal.

The variation of gain with the synaptic weight voltage manifests itself in the switching characteristics of the amplifier's output. On the leading edge of an input pulse the transconductance stage is switched in. As high weight voltages give small feedback loop gains and thus large phase margins, the response of the system to the leading edge of a pulse is slower than for a low weight voltage. On the trailing edge however, as the

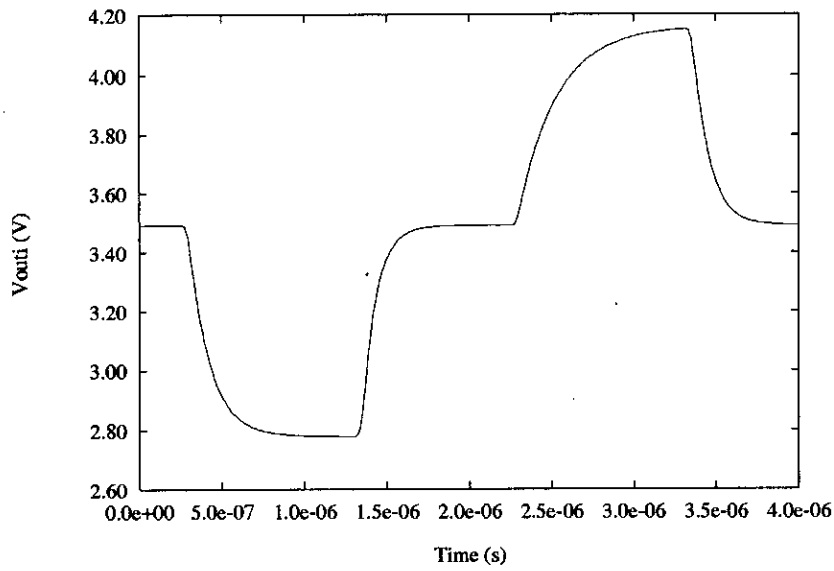


Figure 4.14 HSPICE Simulation of Feedback Amplifier's Response to a Pulse Stream for V_{Tij}=2.0V and V_{Tij}=5.0V.

transconductance stage has now has been switched out, the response is now solely determined by the buffer stage. Thus the leading and trailing edges of a pulse have different responses. The simulation results in Figure 4.14 show these effects.

In determining the size of the compensation capacitor, a worst case feedback loop gain of 0.4 was used. Therefore the compensation capacitor can be reduced by a factor of 2.5.

The operational amplifier was designed for capacitive loads of up to 20 pF. The above feedback gain will allow loads of up to 50 pF to be driven before the operational amplifier starts to become unstable. As the estimated capacitive load for 100 synapses is about 5 pF there is a useful safety margin.

4.3.2. A Voltage Integrator

The voltage integrator comprises a differential stage and cascode current mirrors (Figure 4.15). Current, I_{EXT}, for the current mirrors is determined off-chip to minimise the effects of process variation on the integrator's output current range.

The differential amplifier steers currents down the two paths, M1/M3A/M3B and M2/M4A/M4B, according to the voltage difference between V_{outi} and reference voltage V_{OZ}. When these 2 inputs are at the same voltage, the current through transistors M3A/M3B ($I_{M3A/M3B}$) equals the current through M4A/M4B ($I_{M4A/M4B}$). Thus $I_{M4A/M4B}$

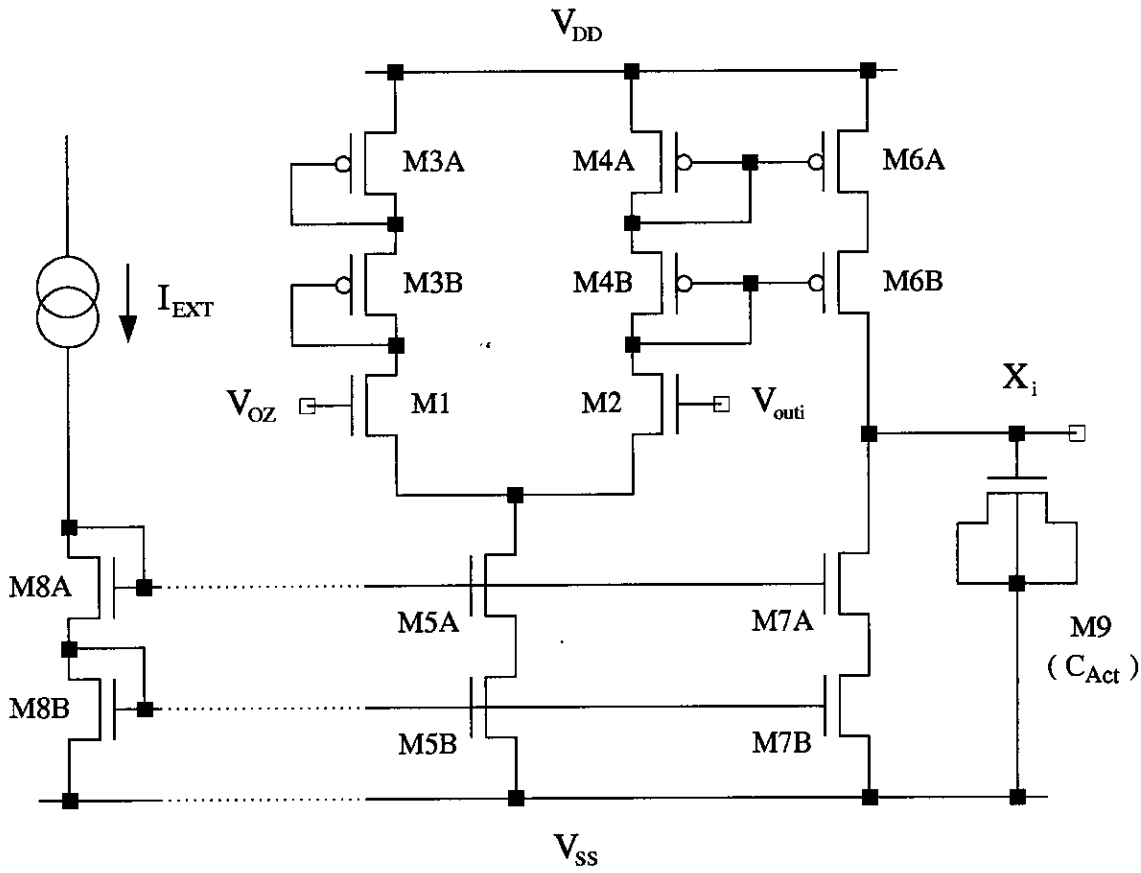


Figure 4.15 A Voltage Integrator Based on a Differential Amplifier and Cascode Current Mirrors.

is half the value of $I_{M5A/M5B}$. At this point the current being supplied to the integration capacitance should balance the current being removed. Thus $I_{M7A/M7B}$ must be half $I_{M5A/M5B}$, so that the net current to the integration capacitor is zero.

As both transistors M1 and M2 are in their saturation regions of operation, the actual voltage to current relationship is sigmoidal. However, by reducing the gain of the differential stage, the linearity over the required input range can be improved. As the integrator capacitor has been implemented as a NMOS transistor, any variations in the gain of the differential stage are tracked by the variations in the integration capacitance. Thus the rate of change of voltage will remain approximately the same over all process variations.

To prove the process tolerance of the voltage to dX_i/dt transfer characteristic, a more stringent analysis is required. First let us assume that M1 and M2 are matched transistors and are both operating in the saturation region. Allen and Holberg [116] give the important equations for the differential stage as

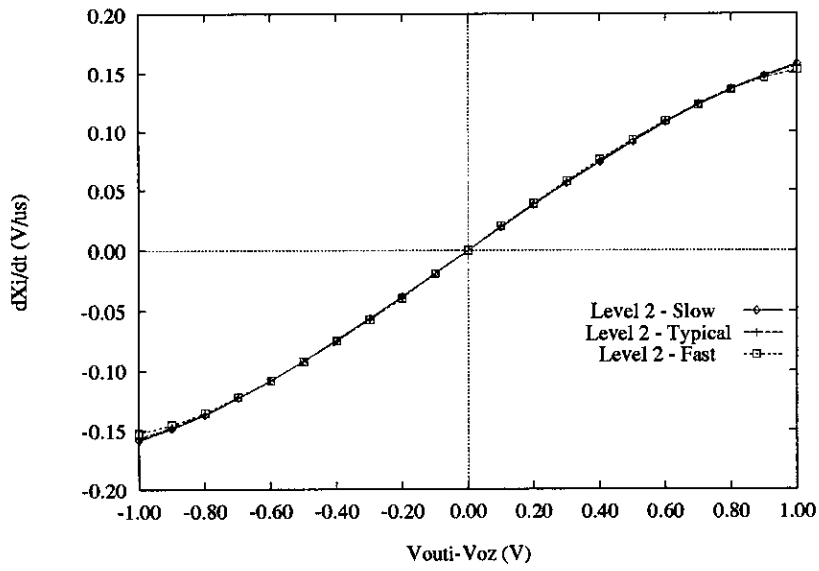


Figure 4.16 Simulated Process Variation for the Voltage Integrator.

$$V_{Diff} = V_{outi} - V_{OZ} = \left(\frac{2I_{M2}}{\beta_{M2}}\right)^{\frac{1}{2}} - \left(\frac{2I_{M1}}{\beta_{M1}}\right)^{\frac{1}{2}} \tag{4.33}$$

and

$$I_{Tail} = I_{M1} + I_{M2} \tag{4.34}$$

Substituting Equation 4.34 into Equation 4.33 and solving for I_{M2} in terms of V_{Diff} gives

$$I_{M2} = \frac{I_{Tail}}{2}\left(1 + V_{Diff}\left(\frac{\beta_{M2}}{I_{Tail}} - \frac{\beta_{M2}^{\,2}}{I_{Tail}^{\,2}}\,\frac{V_{Diff}^{\,2}}{4}\right)^{\frac{1}{2}}\right) \tag{4.35}$$

By using the cascode current mirror relationships the integrator output current, I_i, can be expressed in terms of I_{M2} and I_{Tail}.

$$I_i = I_{M6} - I_{M7} = \frac{\beta_{M6}}{\beta_{M4}} I_{M2} - \frac{\beta_{M7}}{\beta_{M5}} I_{Tail} \tag{4.36}$$

The overall input/output transfer function is given by combining Equations 4.35 and 4.36.

$$I_i = I_{Tail}\left[\frac{\beta_{M6}}{\beta_{M4}}\frac{1}{2}\left(1 + V_{Diff}\left(\frac{\beta_{M2}}{I_{Tail}} - \frac{\beta_{M2}^{\ 2}}{I_{Tail}^{\ 2}}\frac{V_{Diff}^{\ 2}}{4}\right)^{\frac{1}{2}}\right) - \frac{\beta_{M7}}{\beta_{M5}}\right] \tag{4.37}$$

For the voltage integrator to be correctly balanced, I_i must be zero when V_{Diff} equals zero. This leads to the following condition on the above β ratios.

$$2\frac{\beta_{M7}}{\beta_{M5}} = \frac{\beta_{M6}}{\beta_{M4}} = R_\beta \tag{4.38}$$

Therefore

$$I_i = \frac{I_{Tail}R_\beta V_{Diff}}{2}\left(\frac{\beta_{M2}}{I_{Tail}} - \frac{\beta_{M2}^{\ 2}}{I_{Tail}^{\ 2}}\frac{V_{Diff}^{\ 2}}{4}\right)^{\frac{1}{2}} \tag{4.39}$$

Differentiating Equation 4.39 with respect to V_{Diff} and then setting V_{Diff}=0V gives the differential gain of the voltage integrator at its mid point as

$$\frac{dI_i}{dV_{Diff\,(V_{Diff}=0V)}} = R_\beta\left(\frac{\mu_o\varepsilon_{OX}}{t_{OX}}\frac{W_{M2}}{L_{M2}}\frac{I_{Tail}}{4}\right)^{\frac{1}{2}} \tag{4.40}$$

where $\beta_{M2} = \dfrac{\mu_o\varepsilon_{OX}}{t_{OX}}\dfrac{W_{M2}}{L_{M2}}$

This allows the rate of dV_i/dt for the activity capacitor to be found

$$\frac{dV_i}{dt}_{(V_{Diff}=0V)} = \frac{1}{\dfrac{\varepsilon_{OX}W_{Act}L_{Act}}{t_{ox}}}R_\beta\left(\frac{\mu_o\varepsilon_{OX}}{t_{OX}}\frac{W_{M2}}{L_{M2}}\frac{I_{Tail}}{4}\right)^{\frac{1}{2}} \tag{4.41}$$

where

$\dfrac{\varepsilon_{OX}W_{Act}L_{Act}}{t_{ox}}$ = Capacitance of the transistor acting as the activity capacitor(F)

Re-arranging gives the gain of the dV_i/dt characteristic for V_{Diff}=0V as

$$\frac{dV_i}{dt}_{(V_{Diff}=0V)} = \frac{1}{W_{Act}L_{Act}}R_\beta\left(\frac{\mu_o t_{OX}}{\varepsilon_{OX}}\frac{W_{M2}}{L_{M2}}\frac{I_{Tail}}{4}\right)^{\frac{1}{2}} \tag{4.42}$$

The process parameters in Table 4.2 show that the tolerances of the fabrication process cause the gate oxide thickness, t_{OX}, and the channel mobility, μ_O, to vary in opposite directions. For example in the Level 2 Slow transistor model t_{OX} is 5% bigger than normal while the value of μ_O is 6% smaller. Due to the division in Equation 4.40, the variation of these values is amplified while in Equation 4.42 the multiplication results in the variations approximately cancelling each other out, confirming the hypothesis made about

the variations in the integration capacitance tracking the variations in the gain of the differential stage.

	$dI_i/d(V_{outi} - V_{OZ})$ (μA/V)			% Variation
	Slow	Typical	Fast	
Theory	0.629	0.663	0.694	+4.7%/-5.1%
HSPICE	0.548	0.586	0.620	+5.8%/-6.5%

Table 4.6a Percentage Variation in the Gain of the Voltage Integrator's Voltage to Current Characteristic. All percentages are taken relative to the typical responses.

C_{Act} (pF)			% Variation
Slow	Typical	Fast	
2.896	3.031	3.183	+5.0%/-4.5%

Table 4.6b Percentage Variation of the Activity Capacitance, C_{Act}.

$V_{outi} - V_{OZ}$ (V)	dX_i/dt (V/μs)			% Variation
	Slow	Typical	Fast	
-1.00	-0.158	-0.156	-0.152	+2.5%/-1.7%
-0.50	-0.092	-0.092	-0.092	+0.4%/-0.1%
0.00	0	0	0	+0%/-0%
0.50	0.091	0.093	0.093	+0.2%/-1.0%
1.00	0.158	0.156	0.153	+1.1%/-2.6%

Table 4.6c Percentage Variation for the Voltage Integrator Rate of dv/dt. All percentages are taken relative to the mean response for $V_{outi} - V_{OZ} = -1.00$V.

The results of the HSPICE Level 2 simulations to confirm this theoretical result are summarised in Tables 4.6a, 4.6b and 4.6c. HSPICE gives the variations in the voltage integrator's current gain and the activity capacitance as both ±5% (Tables 4.6a and 4.6b). Thus the variations should cancel. Over the middle section of the input range ($-0.5V<(V_{outi} - V_{OZ})<+0.5V$) the tracking of the variations is very good, reducing the variance of the results for the dX_i/dt characteristic to less than 1% (Figure 4.16 and Table 4.6c). At the extreme ends of the input range, the nature of the differential stage causes

the output current to tend towards a limit (Figure 4.16). Thus a change in the differential gain will not significantly alter the output current and the variations in output current are no longer cancelled out by the capacitance variations.

Table 4.7 compares the linearity of the Level 2 typical response with that of its best fit straight line over the maximum range for $(V_{outi} - V_{OZ})$. The worst case variation is ±4.6% at the extremes of the response. However as the magnitude of $(V_{outi} - V_{OZ})$ becomes smaller, the match between the linear best fit line and the actual response improves greatly.

The standard operational amplifier integrator circuit would also have been suitable, but the need for a second operational amplifier would render the resultant neuron far too big and power-hungry. Thus a voltage integrator based on a differential stage offers a good compromise between performance, size and power consumption.

$V_{Tij} - V_{OZ}$ (V)	I_i (μA) Typical	Linear Best Fit	% Variation
-0.800	-0.412	-0.432	+4.6%
-0.500	-0.280	-0.270	-2.3%
0.000	0.000	0.001	-0.0%
0.500	0.281	0.271	+2.3%
0.800	0.415	0.433	-4.6%

Table 4.7 The Linearity of the Voltage Integrator.

4.3.3. A Voltage Controlled Oscillator

To form a complete neural network the integrated output of the voltage integrator is used to control the duty cycle of a Voltage Controlled Oscillator (VCO). The VCO for this system was designed by Alister Hamilton [80]. The VCO outputs a pulse stream of fixed width pulses at a frequency determined by its neural activity input. For this work, the neural state represented by the PFM signal produced by the VCO is specified as a duty cycle, DC_i, rather than as a pulse frequency.

The naturally non-linear transfer characteristic of the differential stage within this VCO design, allows the duty cycle to vary from 0 to 50% as a sigmoidal function of the input activity voltage. A process invariant pulse width is achieved by using a Phase Locked Loop and an external reference clock to determine the current, I_h (Figure 4.17). Current I_l determines the inter-pulse spacing. It is this current which is controlled by the differential transconductance stage.

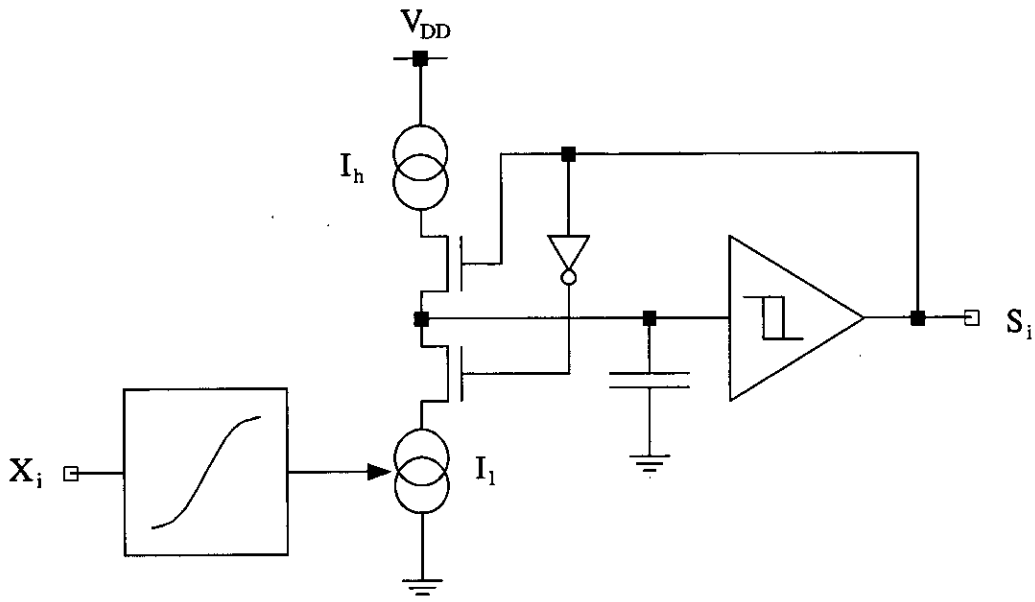


Figure 4.17 Schematic of the Voltage Controlled Oscillator.

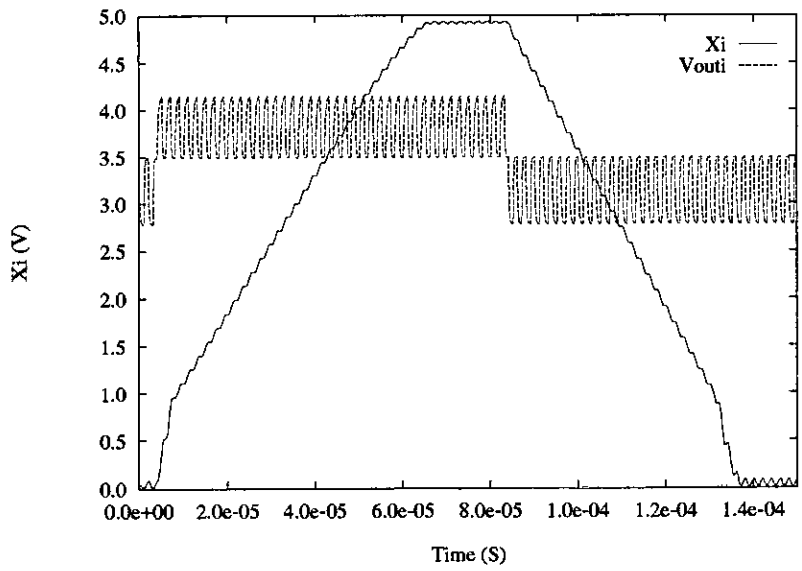


Figure 4.18 A HSPICE Level 2 Typical Simulation of Full Excitation and Inhibition.

4.4. The Complete System

Figure 4.18 shows a HSPICE simulation of the distributed feedback synapses and the voltage integrator working together. For the initial part of the simulation the synapse has a fully excitatory weight (5V) and a 500kHz, 1μs pulse stream (50% duty cycle) applied to it. After 90μs the weight is changed to fully inhibitory (2V). This change is clearly shown by the output of the feedback operational amplifier, V_{outi}, swinging from its upper to lower limit.

There are a few points worth noting here. The very fast rates of change in X_i below 1V occur because the transistor which is being used as an integration capacitor does not reach the desired capacitance until it is fully switched on. This means that the voltage across the transistor must be at least one threshold voltage. Again the imbalance between the magnitude of the response to fully inhibitory and excitatory weights is illustrated by the slope of the X_i trace for the inhibitory weight, which is greater than that for the excitatory weight.

4.5. Conclusions

The 3-transistor transconductance synapse has two main problems. Firstly, its output current is subject to the full effects of process variations. Secondly, the cascadability of a system built around it is poor, as the operational amplifier in the current integrator must supply large currents and drive large activity capacitors. The development of the distributed feedback synapse solved these problems while still retaining the transconductance multiplier's twin advantages of linearity and compactness. These additional virtues of process invariance and cascadability are the keys to the construction of massively parallel arrays of uniform synaptic multipliers.

The mixed signal nature of the synapse multiplication plays a significant role in achieving process invariance. The use of the neural state represented by a pulse stream, to switch the transconductance multiplier current, means that the duty cycle of the resultant current pulses is process invariant, because the duty cycle of the input pulse streams maps directly to the duty cycles of the current pulses.

Comparing the voltages integrator's power consumption of 15μW with the 1-10mW consumption of a standard operational amplifier "leaky" integrator illustrates its very low power consumption. Thus while the voltage integrator may be linear only to within 5% at the extremes of its response, it offers the best compromise between performance, area, power consumption and process invariance.

The addition of the final component in the pulse stream system, the VCO designed by Alister Hamilton, creates an analogue CMOS neural network implementation which is both cascadable and admirably process tolerant.

Chapter 5 goes on to describe the implementation of these circuits in silicon and the results obtained from the testchip.

Chapter 5

SADMANN 1010PI

The previous chapter outlined the design of circuits for a process invariant synapse and a voltage integrator. This chapter discusses the implementation of the SADMANN 1010PI device† fabricated to prove these circuits. A comparison is carried out between the simulation results presented in chapter 4 and measurements from SADMANN.

5.1. Design of SADMANN

The SADMANN device was laid out using a combination of the Magic custom layout software and ES2's (European Silicon Structures) Solo 1400 silicon compiler tools. The custom layout tool was used to create an optimised layout, both in terms of area and the matching of transistors, for the synapse and neuron circuits. Standard cell libraries within Solo 1400 were used to add the necessary address decoding logic and pads to the custom layout to complete the integrated circuit.

In analogue VLSI circuits, the detailed layout of the circuit is vital to the final performance of the fabricated circuit. Both the feedback operational amplifier and the voltage integrator used standard layout techniques to improve the match between important pairs of transistors. These are :

1 As the synapse is sensitive to the value of the reference voltage, V_{Ref} (Figure 4.7), the input transistors in the differential stage of the operational amplifier (M1, M2 in Figure 4.12) were cross-coupled to minimise the voltage offset between the input terminals.

2 To minimise the possible mis-match in reduction ratio of the cascode current mirrors within the voltage integrator, the mirroring transistors were implemented as multiples of the same transistor. This eliminates errors caused by variations in the transistor's aspect ratio, and gives a more constant current ratio.

SADMANN contained three separate pieces of work:

1 A small test array for the distributed feedback synapse described in the last chapter.

† Steve's, Alister's and Donald's Monolithic Analogue Neural Network, 10 output neurons, 10 input neurons, Process Invariant

2 A neural array based on a "self-depleting" neuron [117].

3 An inter-chip self-timed communication scheme loosely based on RS-232 protocols [118].

The latter 2 pieces of work were carried out by Stephen Churcher [117] and Alister Hamilton [118], respectively. Each of the 2 test networks contained a 10 by 10 synaptic array plus 10 output neurons. The resultant 7.5mm by 7.5mm testchip was fabricated using ES2's (European Silicon Structures) ECDM20 2μm digital CMOS process.

The sizes of the synapse, operation amplifier, voltage integrator and VCO cells for the distributed feedback synapse system are shown in Table 5.1. The size of the synapse, while small, is non-optimal due to the need to pitch match to a larger synapse in the second neural test array contained in the SADMANN device. A better indication of the size of the distributed feedback synapse is given by the 200μm by 100μm double synapse cell, in the successor to SADMANN, EPSILON, described in Chapter 6.

Cell	Size
Synapse	165μm x 130μm
Feedback Op-amp	165μm x 250μm
Integrator	165μm x 200μm
VCO	165μm x 165μm

Table 5.1 Cell Sizes for Synapse and Neuron Circuits.

At this point it should be emphasised that SADMANN was only designed to work with PFM signals.

Sections 5.2 and 5.3 describe static and dynamic measurements obtained from SADMANN while Section 5.4 discusses the systematic variations found in the synaptic characteristics.

5.2. Static Measurements

A simple test board was constructed to perform the initial static tests of the SADMANN. No digital support circuitry was included in an effort to reduce noise on the board and thus increase the accuracy of the measurements. To this end the clock signal required to clock the weights in was provided by an external pulse generator.

All 10 devices received from ES2 powered up correctly without any latch-up problems. However, on one chip, the multiplexer for monitoring the feedback operational amplifier outputs and activity capacitor voltages failed. The performance of a second

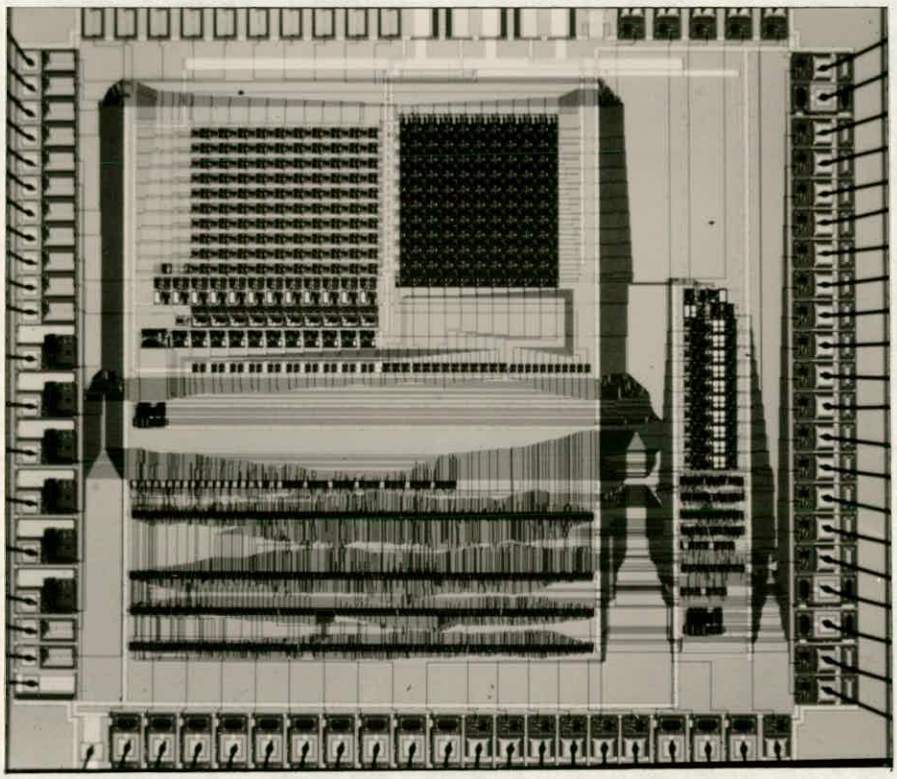


Figure 5.1 Silicon Layout of Testchip.

chip was significantly altered after 20V had been inadvertently applied to it due to a faulty ground connection. Thus only 8 of the 10 chips were suitable for detailed testing.

5.2.1. Automatic Bias Circuits

The first test performed on SADMANN was to determine if the V_{OZ} and V_{SZ} feedback loops (Figure 4.9) were supplying the correct bias voltages to the neural array. A comparison of the statistics for the measurements of V_{OZ} and V_{SZ} in Table 5.2 with the worst case Level 2 HSPICE simulation values (Table 5.3), yielded some interesting conclusions.

For the V_{OZ} and V_{SZ} feedback loops, the measured and HSPICE results differ by less than 100mV. This was a very good match considering the limited accuracy of the Level 2 HSPICE model. The small (±0.5%) variations in the measured bias values indicated that the circuits correctly compensated for process variations.

To determine whether the bias voltages were correct for the synapse array additional testing was required. With the transconductance stage switched out, the operational

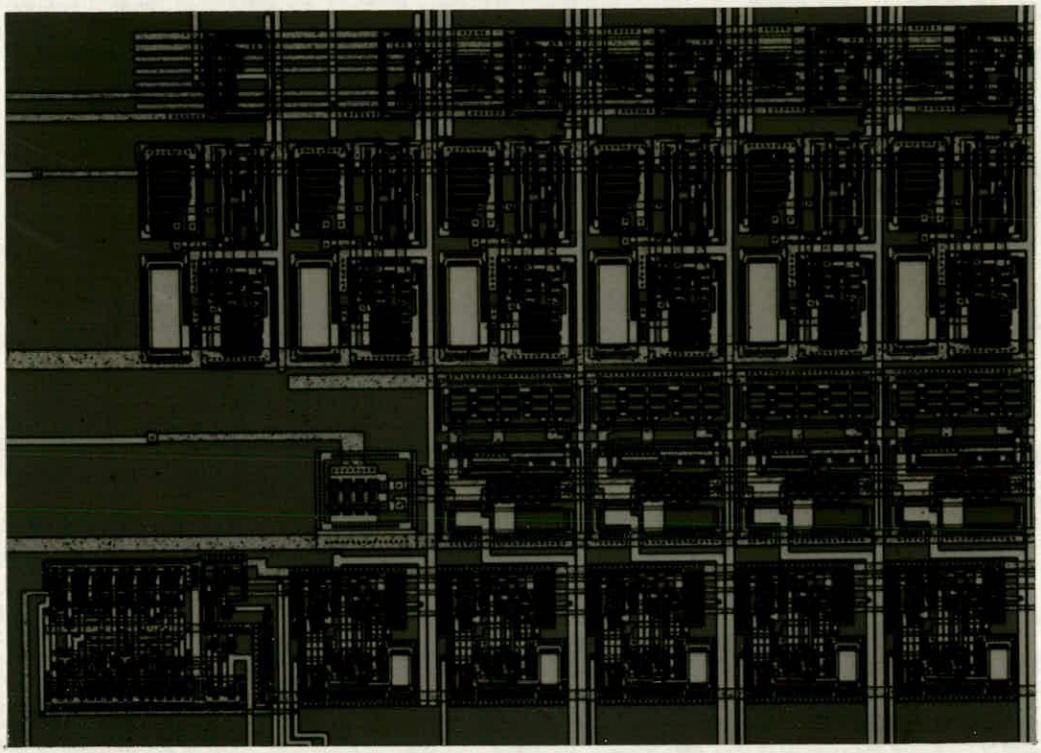


Figure 5.2 Silicon Layout of Synapse and Neuron.

amplifier output was expected to settle around V_{OZ} for that chip and indeed it did. The change in the operational amplifier's output as the transconductance stage was switched in and out (with $V_{Tij}=V_{TijZ}$) was expected to be zero, provided the V_{SZ} feedback loop functioned. Unfortunately, this was not the case. To achieve the correct balance V_{TijZ} had to be raised for all chips by 70mV to 3.57V. This slight mis-match may have been caused by temperature differences created by a non-optimal cell placement. The cell which determines V_{SZ} lies at the bottom left corner of the synaptic array and thus does not experience the same heating effect as synapses deep in the array. Another possibility is that due to the effects of averaging, the output of a feedback circuit based on a single transconductance stage may not be representative of the mean response of 100 synapses. The systematic nature of the problem would tend to suggest that the former explanation is correct.

However, neither of these explanations accounts for the fact that the V_{OZ} feedback loop works while the V_{SZ} feedback loop does not, despite the feedback loops having very similar circuits. This points to a mis-match between the V_{SZ} cell and the synapse cell,

	V_{OZ} (V) (V_{Bias} = 2.50V)	V_{SZ} (V)	
		V_{TijZ} = 3.50V	V_{TijZ} = 3.57V
Minimum	3.52	4.56	4.61
Mean	3.53	4.57	4.63
Maximum	3.54	4.60	4.67
Variation	±0.3%	+0.7%/-0.2%	+0.9%/-0.4%
Standard Deviation	±0.2%	±0.3%	±0.4%

Table 5.2 The Percentage Variations for V_{OZ} and V_{SZ} Bias Voltages. All percentages are taken relative to the mean responses.

Process	V_{OZ} (V) (V_{Bias} = 2.50V)	V_{SZ} (V)	
		V_{TijZ} = 3.50V	V_{TijZ} = 3.57V
Level 2 - Slow	3.51	4.52	4.59
Level 2 - Typical	3.49	4.49	4.56
Level 2 - Fast	3.47	4.47	4.54
Variation	+0.6%/-0.5%	+0.5%/-0.5%	+0.5%/-0.6%

Table 5.3 The Simulated Variations in V_{OZ} and V_{SZ} Bias Voltages. All percentages are taken relative to the mean responses.

caused by the non-identical layout around the transconductance multipliers.

This mis-matching can be prevented by making the automatic bias cells exact replicas of the synapses. The accuracy of the bias voltages would also be improved by having a column of feedback cells **in the middle** of the synaptic array to take into account the effects of averaging and temperature.

V_{DD}=1.60V	V_{Ref}=0.80V	V_{TijZ}=3.57V
V_{SS}=0V	V_{Bias}=2.50V	0V<V_{Tij}<5.00V

Table 5.4 The Voltages applied to the Synapse for the Static and Dynamic Measurements.

5.2.2. Synapse

The output swing of the operational amplifier for a single sweep of the synaptic weight from 0V to 5V is less than 200mV. It is therefore very difficult to measure the characteristics of a single synapse accurately. Thus the static results presented in this section are for a column of 10 synapses working in parallel.

To give a common base for the measurements, the reference voltages in Table 5.4 were applied to all chips. The aim was to test the process tolerance of the synapse circuit in association with the automatic bias feedback loops. All of the pulse stream inputs were switched to 5V (ON), and the output of a synaptic column was taken as the voltage difference ($V_{outi} - V_{OZ}$).

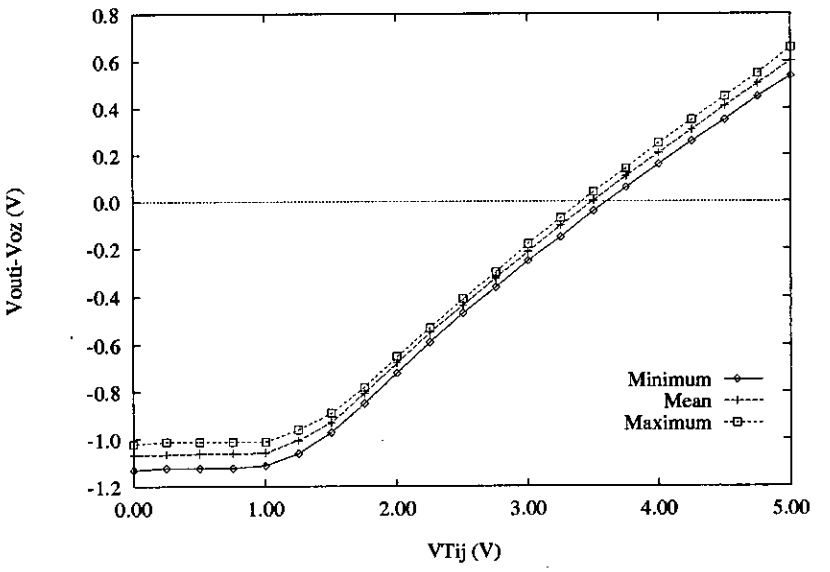


Figure 5.3 The Variation in the Static Measurements of the V_{Tij} to $(V_{outi} - V_{OZ})$ characteristic.

Figure 5.3 shows the worst case variation for the post-fabrication measurements from 80 synaptic columns (8 chips × 10 synaptic columns). This graph confirms the linear relationship between the synaptic weight voltage and the output voltage of the operational amplifier. Comparing the results in Table 5.5 with that of the HSPICE simulations of the synapse (Table 4.4) demonstrates a close agreement between theory and reality. The maximum variation in operational amplifier output voltage over the linear weight range as a fraction of the total operational amplifier output range was found to be about ±10% (Table 5.5). This result was very encouraging and it compared very favourably with the 20% to 50% variation in performance quoted for simple current mirrors fabricated using the MOSIS digital process [83, 119, 66].

V_{Tij} (V)	$(V_{outi} - V_{OZ})$ (V)				
	Minimum	Mean	Maximum	Variation	Std Dev
0.00	-1.13	-1.06	-1.02	+7.5%/-10.8%	±3.6%
1.00	-1.11	-1.06	-1.01	+7.8%/-8.8%	±3.5%
2.00	-0.72	-0.68	-0.65	+4.5%/-7.1%	±2.3%
3.00	-0.25	-0.21	-0.18	+5.5%/-6.1%	±2.4%
4.00	0.16	0.21	0.25	+7.0%/-7.9%	±2.8%
5.00	0.54	0.60	0.66	+9.3%/-10.6%	±3.4%

Table 5.5 The Percentage Variations of the V_{Tij} to $V_{outi} - V_{OZ}$ Characteristic. All percentages are taken relative to the mean response for V_{Tij}=5.00V (0.60V). (The full set of results can be found in Table A4.1 in Appendix 4)

5.2.3. Voltage Integrator

During the design of SADMANN, provision was made for the output currents of the voltage integrators in neurons 0, 1 and 2 to be monitored. Current-to-voltage converter circuits, based on external operational amplifiers, were used to convert the integrators' output currents into values which could be measured directly.

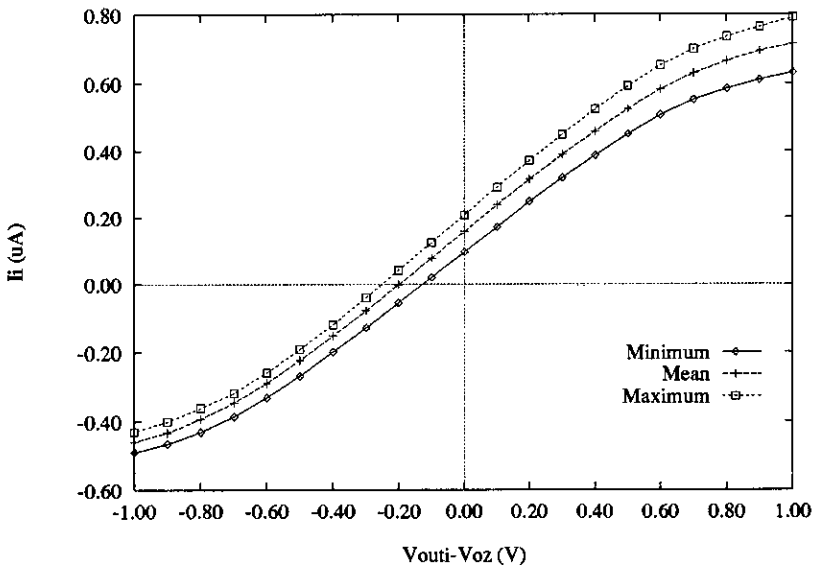


Figure 5.4 The Variation in the Output Current, I_i, of the Voltage Integrator.

$(V_{outi} - V_{OZ})$ (V)	I_i (μA)				
	Minimum	Mean	Maximum	Variation	Standard Deviation
-1.00	-0.49	-0.46	-0.43	+6.3%/-6.6%	±3.2%
-0.40	-0.20	-0.15	-0.12	+7.2%/-10.0%	±4.1%
0.00	0.10	0.16	0.21	+10.9%/-13.5%	±5.4%
0.40	0.39	0.46	0.52	+14.2%/-15.7%	±6.9%
1.00	0.63	0.71	0.79	+16.8%/-18.1%	±7.8%

Table 5.6 The Percentage Variations of the $V_{outi} - V_{OZ}$ to I_i Characteristic. All percentages are taken relative to the mean response for $(V_{Tij} - V_{OZ})$=-1.00V (-0.46μA). (The full set of results can be found in Table A4.2 in Appendix 4)

Figure 5.4 shows the worst case measurements from the 24 voltage integrators (3 integrators × 8 chips) tested. Comparison with Figure 4.16 shows that there was a substantial difference between the measured results and the simulations for the voltage integrator. This is due to a mis-match in the M4/M6 current mirror (Figure 4.15) from the differential stage to the integration capacitor, increasing the output of the current mirror by 25%. Unfortunately no definite explanation has been found to account for this mismatch problem. Since the circuit extracted from the layout simulates satisfactorily, it appears that there is a problem with this particular layout of the circuit rather than a problem with the circuit itself. The voltage integrator effectively has an offset voltage of about 200mV which needs to be compensated for.

The increase in the variation of the output current from ±6% at $(V_{outi} - V_{OZ})$=-1.00V to ±18% at $(V_{outi} - V_{OZ})$=1.00V (Table 5.6), is due to the cumulative effect of 3 separate sub-circuits within the voltage integrator:

1 The M8/M5 current mirror (Figure 4.15) providing the balance current.

2 The M1 and M2 differential input transistors.

3 The M4/M6 current mirror copying the differential stage output current onto the activity capacitor.

For large negative voltage differences, the integrator output current is due solely to the current mirror providing the balance current. Thus the ±6% variation is the result of the effects of process variation on this current mirror. As V_{outi} increases, the contributions from the differential stage and the second current mirror become more significant.

At the same time these stages introduce their own additional variations into the integrator's output current. When ($V_{outi} - V_{OZ}$) reaches +1.00V both current mirrors contribute a ±6% variation. With the variation in the differential stage gain supplying a further ±6%, the overall variation is ±18%. The implication is that using a current mirror introduces a ±6% variation in current due to non-uniformities in the fabrication process across a chip.

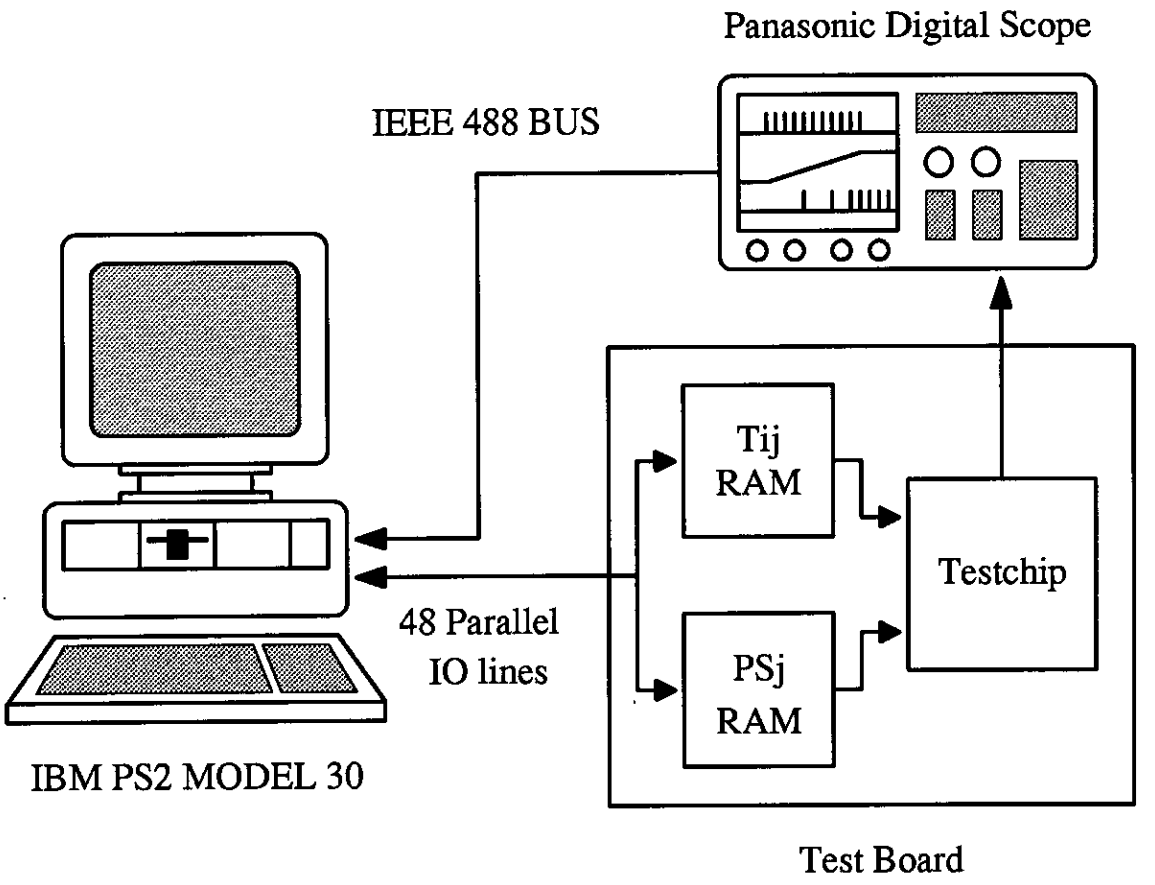


Figure 5.5 The Test System for SADMANN.

5.3. Dynamic Measurements

After the static tests had confirmed the operation of SADMANN, a more sophisticated test board was built. At the heart of this automated test system (Figure 5.5) was an IBM PS2 Model 30 containing parallel IO and IEEE 488 interface cards, through which it controlled the test board and digital storage oscilloscope respectively. The following sequence is initiated by the IBM PS2 to measure the output of a synaptic column:

1 Transfer weight and state information down to the test board via parallel IO card.

2 Select activity voltage to be monitored using the analogue multiplexer in SADMANN.

3 Fire state information into the synaptic array while the digital storage oscilloscope captures the response of the activity capacitor.

4 Apply linear regression to the captured signal to determine the activity capacitor's rate of change, dX_i/dt.

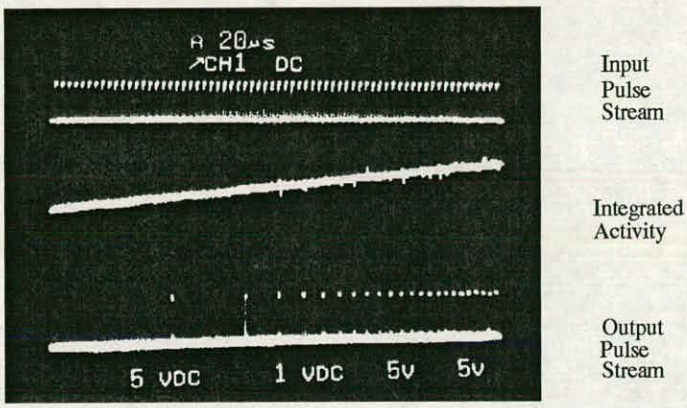


Figure 5.6 An Oscilloscope Photograph of the System in Operation.

Using this system it proved possible to characterise the response of individual synapses, but not without complication. The voltage offset problems of the voltage integrator, coupled with the inability to reset the activity capacitors to a known voltage made initialising the system to a known state difficult. The lack of a reset mechanism was an oversight at the design stage. The presence of a reset capability would have eased the integration of the testchip into the system. Also, the ability to disconnect the activity capacitor from the output stage of the integrator after the completion of a calculation would have prevented the activity voltage from drifting, due the integrator offset.

However these small obstacles were overcome. The voltage integrator has an offset of about 200mV. To compensate for this offset the output of the feedback operational amplifier has to be shifted downwards by 200mV. Each synapse can only generate a shift of approximately ±60mV. Thus 4 synapses in each synaptic column were required as biases to compensate for the voltage integrator's offset. The pulse inputs to these 4 synapses were permanently switched on. The calculation of the bias values required for each neuron was performed automatically by the software written for the test system. While this automatic bias procedure did allow the prototype system to run more or less as desired, it did not completely eliminate the problems caused by current mis-matches

within the voltage integrator.

The operation of a synaptic column and its neuron is illustrated in the oscilloscope photographs of Figure 5.6. The top trace shows a pulse stream arriving at the synapse inputs. The synaptic weights are fully excitatory. The integrator output, shown in the second trace, increases linearly with time.

As the integrator output, and therefore the neuron activity increases the neuron switches on, outputting a pulse stream shown in the third oscilloscope trace.

The upper frequency of the pulse input signals was constrainted by the limited band-width of the operational amplifier used in the synapse feedback circuit. Therefore the input pulse width was set to 1μs and the maximum pulse frequency limited to 500kHz.

The capacitance of the current monitor lines on the outputs of the voltage integrators in neurons 0, 1 and 2 gave them a slower response than neurons 3 to 9. This, in addition to the use of 4 synapses per column as biases, limited the number of synapses characterised on each chip to 42. With 8 chips tested this gave 336 measurements for each combination of weight voltage and duty cycle, which provided sufficient information to allow a conclusion to be reached regarding the process invariance of this neural system.

The variation figures quoted in the text and tables are all expressed relative to the mean rate of dX_i/dt of a pulse stream with a duty cycle of 50% and V_{Tij} = 5.00V (4.95 V/ms, Table 5.7). It should be emphasised that this choice is fairly arbitrary. For example choosing a duty cycle of 50% and V_{Tij} = 1.99V (dX_i/dt= -5.93V/ms, Table 5.7) as the reference point, would lower the percentage variation figures by 20%.

5.3.1. Measurements of Process Variation

To characterise the system in operation, and to determine the process variation of the synapse/voltage integrator combination, 3 sets of measurements were taken for each synapse. They were as follows

1 A sweep of V_{Tij} from 1.11V to 5.00V for a S_j=50%.

2 A sweep of the S_j from 0% to 50% with V_{Tij}=1.99V.

3 A double sweep where V_{Tij} varied from 1.99V to 5.00V while S_j varied between 0% and 50%.

Once these results had been gathered, a second set of experiments was performed to probe the measurement accuracy of the test system. This was done by re-running the above sweeps for testchip number 9 except that for the first two sweeps 25 samples were taken at each measurement point. For the third sweep, only 10 samples per measurement point were taken due to the long run time required (overnight).

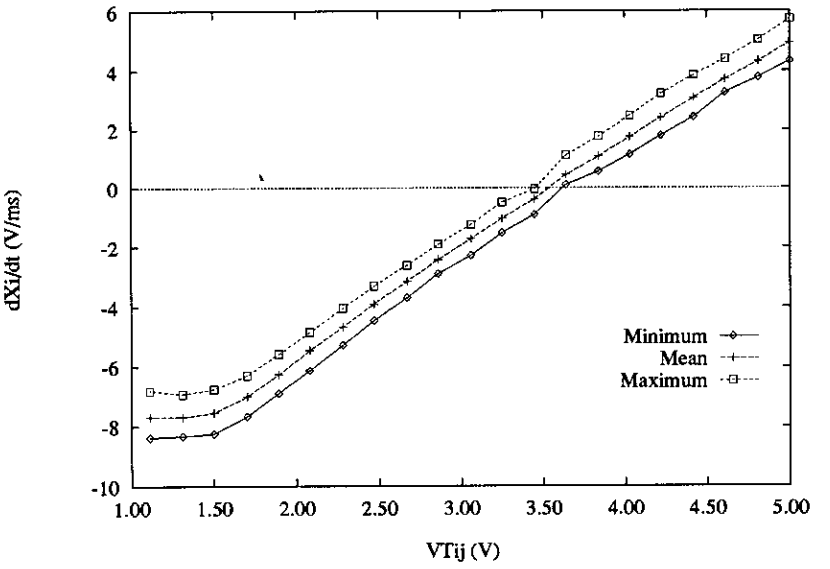


Figure 5.7 Process Variation for Weight (V_{Tij}) Sweep.

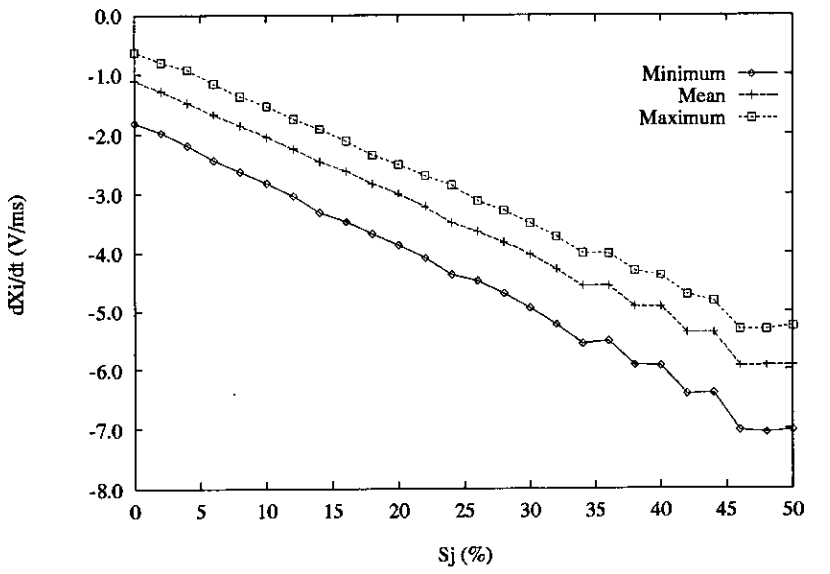


Figure 5.8 Process Variation for Duty Cycle (S_j) Sweep.

Figures 5.7, 5.8 and 5.9 show the synaptic characteristics found. Table 5.7 gives a summary of the percentage variations and Table 5.8 gives the measurement accuracy of the system for selected measurements. Appendix A5, Tables A5.1-A5.4 contain full listings of the results gathered.

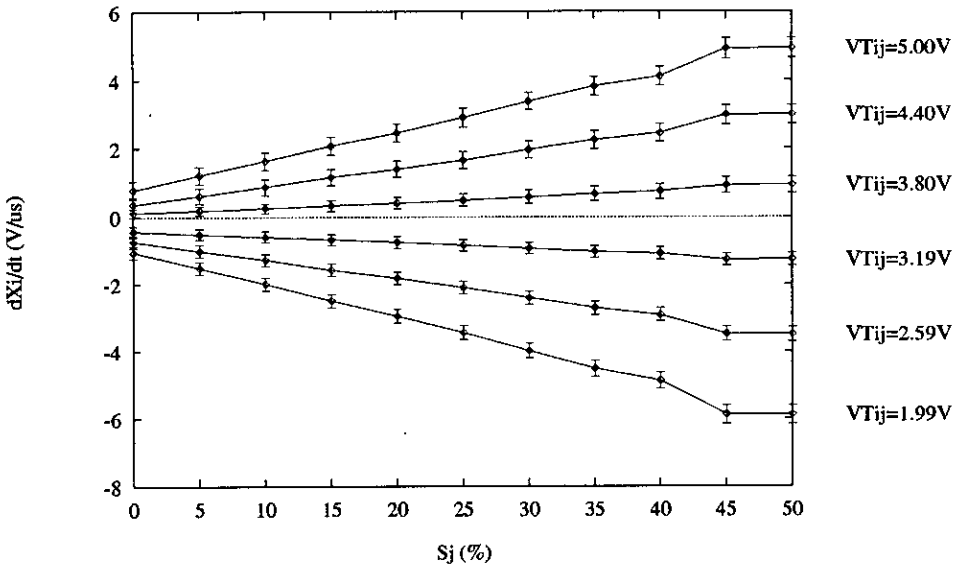


Figure 5.9 Process Variation for Double Sweep.

The linearity of this system is immediately apparent from the characteristics shown in Figures 5.7, 5.8 and 5.9. There are slight non-linearities in these graphs but as is explained below, these are due to system problems rather than to the circuits themselves.

1 The pinch in the traces relating to the weight voltage sweep (Figure 5.7) around the zero weight value was due to the integrator ramping from 0V or 5V. If the integrator was at 0V, and the synapse was slightly inhibitory due to the effects of process variation, the measured dX_i/dt was zero, as it was not possible for the activity voltage to drop below 0V. The ability to reset the activity capacitors to a known value, say 2.5V, would have avoided these problems.

2 The mean response of the duty cycle sweep (Figure 5.8) was expected to pass through the origin because, if the input duty cycle is 0%, ie no pulses, then the rate of dX_i/dt should be zero. As Figure 5.8 shows, the mean value trace was in fact offset by 1 V/ms. The offset was again due to the problems experienced in balancing the voltage integrator correctly. This also applies to the results for the double sweep, Figure 5.9.

3 The step nature of the graphs in Figures 5.8 and 5.9 was due to limitations in the pulse generation hardware rather than in the synapse circuit. As the static RAM chip supplying the pulse signal was clocked at 4MHz, the pulse signal was quantised in steps of 250ns. With this limited resolution, only a few distinct high value (30%-50%) duty cycles could be generated. For example 37%-39% maps to 40%,

Sweep	S_j (%)	V_{Tij} (V)	dX_i/dt (V/ms)				
			Min	Mean	Max	Variation	Std Dev
V_{Tij}	50	5.00	4.35	4.95	5.75	+16.0%/-12.3%	±5.2%
DC_j	50	1.99	-7.02	-5.93	-5.25	+13.6%/-22.2%	±6.1%
Double	50	1.99	-6.56	-5.87	-5.19	+13.6%/-13.9%	±5.5%
	50	5.00	4.16	4.94	5.66	+14.4%/-15.8%	±5.8%

Table 5.7 Selected Variations for the Measurements of dX_i/dt. All percentages are taken relative to the mean response for V_{Tij}=5.00V and S_j=50% (4.95 V/ms).

Sweep	S_j (%)	V_{Tij} (V)	dX_i/dt (V/ms)	
			Variation	Std Dev
V_{Tij}	50	5.00	+4.2%/-6.0%	±1.7%
DC_j	50	1.99	+2.8%/-3.2%	±1.4%
Double	50	1.99	+2.4%/-2.8%	±1.6%
	50	5.00	+3.3%/-4.7%	±1.8%

Table 5.8 The Accuracy of the dX_i/dt Measurements. All percentages are taken relative to the mean response for V_{Tij}=5.00V and S_j=50% (4.95 V/ms).

40%-44% to 44% and 45%-50% to 50%. Subsequent designs will reduce these effects by clocking the system at 10MHz, improving the resolution by a factor of 2.5.

It was, however, pleasing to note that the synapse and integrator circuitry was sensitive enough to reveal the inadequacies of the system around it.

The percentage variations for the 336 synapses characterised in the 8 chips tested are given on Table 5.7. The variations for the worst case measurements were typically about ±15%. The full set of figures in Appendix A5 show that there was significant variation in the worst case variation percentages. This may have been due to the presence of one or two "rogue" measurements. A more consistent measure of the variations can be

found by calculating the standard deviation of the results which in this case is ±6%. Comparing the above variation with the measured static variations for the synapse, ±3.4%, and the voltage integrator, ±5.4% (Vouti – Voz=0V), leads to the conclusion that the variations in the differential gain were tracked by the variations in the activity capacitance, thus decreasing the overall variation.

The reported standard deviation figure for a similar characterisation of the Hamilton pulse stream synapse [118] was ±18.06%, with -40.0%/+44.9% as the worst case variations. A direct comparison with the Hamilton synapse is valid, as it was fabricated on the same ECDM20 2μm process. This comparison verifies that the distributed feedback synapse is indeed relatively tolerant to the effects of process variations. The comparison is further enhanced by the observation that the Hamilton synapse process variation figures are based on the results from only one chip. They do not, therefore, represent the synaptic multiplication accuracy of a multichip system.

The above variations relate to the system as a whole (ie the testchip plus the support circuitry, digital oscilloscope and the analysis software). The second set of experiments mentioned at the beginning of this section measured the accuracy of the test system by taking up to 25 samples for each combination of V_{Tij} and S_j for a particular synapse. The percentage accuracy as a standard deviation was ±2% (Table 5.8).

This indicates that the variation due the circuits themselves is less than the ±6% reported above. Comparing this with the standard deviation of the current produced by the current mirror in the voltage integrator, ±3.2% (Table 5.6) demonstrates that for a system made up of such a large number components, a variation as small as ±6% is very good.

5.3.2. Systematic Variations

Finally, to find systematic variations across the synaptic array, the results for the DC_j=50%, V_{Tij}=2.08V, and DC_j=50%, V_{Tij}=5.00V measurements for a particular synapse were averaged over the 8 chips tested. The 3 dimensional plots of the responses are shown in Figures 5.10 and 5.11. There are two observations to be made here.

1 The variation **within** a synaptic column is small compared with the variation **between** synaptic columns. Thus the matching between synapses is better than the matching between the voltage integrators.

2 The performance of the synapses degrades very slightly, moving across the array from neuron 9 to neuron 3. This degradation in performance is due to the voltage gradient in the power supplies across the array caused by the resistance of the power tracks.

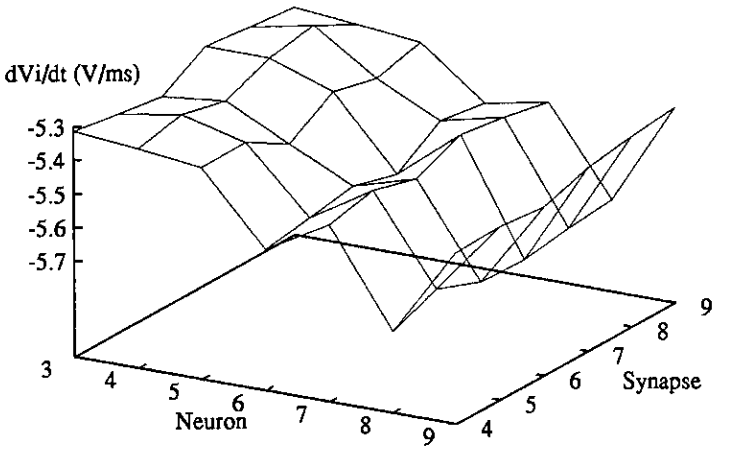


Figure 5.10 Across Chip Variation in the Synaptic Response for V_{Tij}=2.08V and S_j=50%.

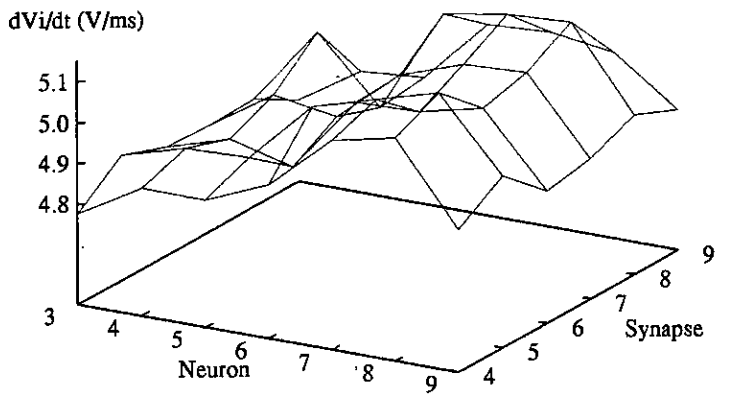


Figure 5.11 Across Chip Variation in the Synaptic Response for V_{Tij}=5.00V and S_j=50%.

5.4. Conclusions

The results of the static and dynamic tests clearly prove that the circuits described in Chapter 4 perform functionally as desired. The performance of the synapse/voltage integrator combination compares well with the measured variations of the cascode current mirrors on the same chip and the reported variations for the Hamilton synapse. The major objective of the design, that of creating circuits which have an in-built resistance to process variations, has been achieved on a fabrication process optimised for digital rather than analogue circuits to the extent that variations due to process have been limited to ±6%.

Several valuable lessons in analogue circuit design have also been learned.

1 The results of the performance tests on the voltage integrator confirm the suspicion that the more components a system includes, the greater are the effects of changes in the process parameters. Thus simple analogue circuits will usually be more process invariant than complex ones, unless special design techniques are used, for instance cancelling the variations in the integrator's gain with the variations in the activity capacitance.

2 While it proved impossible to trace the detailed reason for the voltage integrator's mirroring problem, a definite solution is known in that the circuit can be re-designed with the tail current for the differential stage and the balance current determined separately to ensure that the integrator can always be balanced.

3 HSPICE simulations in which there is perfect transistor matching, are not a reliable guide to the use of feedback loops. In practice it appears that, as the fabrication process is not constant within a chip, the feedback value may not be correct for a synapse which is far away from the feedback circuit. This problem should be eased by placing a column of the feedback cells in the centre of the synapse array. However, if the process changes significantly across a chip, it becomes almost impossible to calculate bias voltages to any degree of accuracy. Under this scenario each synapse would require its own process compensation circuitry.

4 Truly accurate analogue designs can only be developed if creative circuit and layout techniques, such as those described in this chapter, are used in combination with a fabrication process optimised for analogue circuits.

Despite the fact that SADMANN was only designed for PFM signals, the synaptic multiplier circuits are also capable of working with PWM signals. In the successor to SADMANN, EPSILON (described in Chapter 6), this duality is exploited and as a result EPSILON has two distinct operating modes; PFM and PWM.

Chapter 6

EPSILON 30120PI

With the characterisation of the distributed feedback synapse complete, a comparison with other pulse based multipliers is now necessary to find the most appropriate synapse design for the final neural VLSI demonstrator. What follows is a comparison of the salient features of the distributed feedback synapse and the previous pulse stream synapses developed by Brownlow [105-107] and Hamilton [115, 112] (Table 6.1).

	Distributed Feedback Synapse	Brownlow Synapse	Hamilton Synapse
Linear Multiplication	Yes	Yes	Yes
Area	165x130μm^2	65x65μm^2	175x73μm^2
Pulse Modes	PFM & PWM	PFM	PFM
Cascadable	Yes	No	Yes
Process Invariant	Yes	No	No

Table 6.1 A Comparison of 3 Pulse Based Synapse Designs.

1 All three designs possess a linear multiplication characteristic over a wide range of synaptic weight and state values. This compares well with the sigmoidal characteristic of the Gilbert multiplier used in Intel's ETANN device.

2 The relatively large area for the distributed feedback synapse is misleading. The SADMANN layout is non-optimal, since the synapse is pitch-matched to the synapse contained in the second neural test array. A more realistic area estimate for a fully-optimised SADMANN synapse is 100x100μm^2.

3 The Brownlow and Hamilton synapses are fundamentally restricted to operating on PFM signals. The distributed feedback synapse is more flexible as it works equally well with PFM or PWM signals.

4 As the operational amplifier in the integrator needs to redesigned when the number of switch-capacitor synapses is scaled, the Brownlow synapse is the only one of the three which is not easily cascadable.

5 The source follower buffer stages in the Brownlow and Hamilton synapses render both designs process variant. The output of a source follower buffer is directly dependent on the value of the threshold voltage for the process.

While the distributed feedback synapse is not the smallest of the 3 synapses, this shortcoming is amply compensated by its ability to work with different pulse encoding techniques without modification, and by its process invariance. As process invariance is the overriding concern for the Hopfield/Tank and the Kohonen neural networks, the distributed feedback synapse was chosen to form the basis of the full scale VLSI neural implementation.

As intimated at the start of Chapter 2, this implementation was designed to integrate MLP networks as well as the Hopfield/Tank and the Kohonen networks. The specifications for the networks are shown below.

1 A 89-16-2† (89 input neurons, 16 hidden neurons and 2 output neurons) MLP network for finding roads in images [117]. Sigmoidal neuron transfer function.

2 A 100-64-32 MLP network [118]. Sigmoidal neuron transfer function.

3 A Hopfield/Tank network consisting of 100 neurons and 10000 synapses (100x100 array). Sigmoidal neuron transfer function.

4 A Kohonen network consisting of 30 neurons and 90 synapses (30x3 array). Linear neuron transfer function.

Unfortunately, funding restrictions limited the maximum die size to 100mm^2. The 10000 synapses required by the largest of the above four networks, the Hopfield/Tank network for the 10-city TSP, occupies all of the available 10x10mm^2 of silicon, before taking into account the area required the neurons, the weight addressing circuitry and the input/output pads. Thus it was not feasible under this area limitation to implement a network large enough to run the Hopfield/Tank network for the 10-city TSP on a single silicon die.

To solve this problem the generic neural VLSI building block, EPSILON 30120PI‡, was developed. This device was something of a creative compromise. The specifications for EPSILON are shown in Table 6.2. For the Hopfield/Tank and the Kohonen neural networks 4 EPSILON's are linked in parallel to form a 120 by 120 synaptic array. MLP architectures are constructed by cascading rows of EPSILONs in series, each row representing one layer of the MLP.

† This is the original specification for the network. Subsequent work reduced the network size to 45-14-2.

‡ (Edinburgh Pulse Stream Integrated Learning Oriented Network, 30 output neurons, 120 input neurons, Process Invariant)

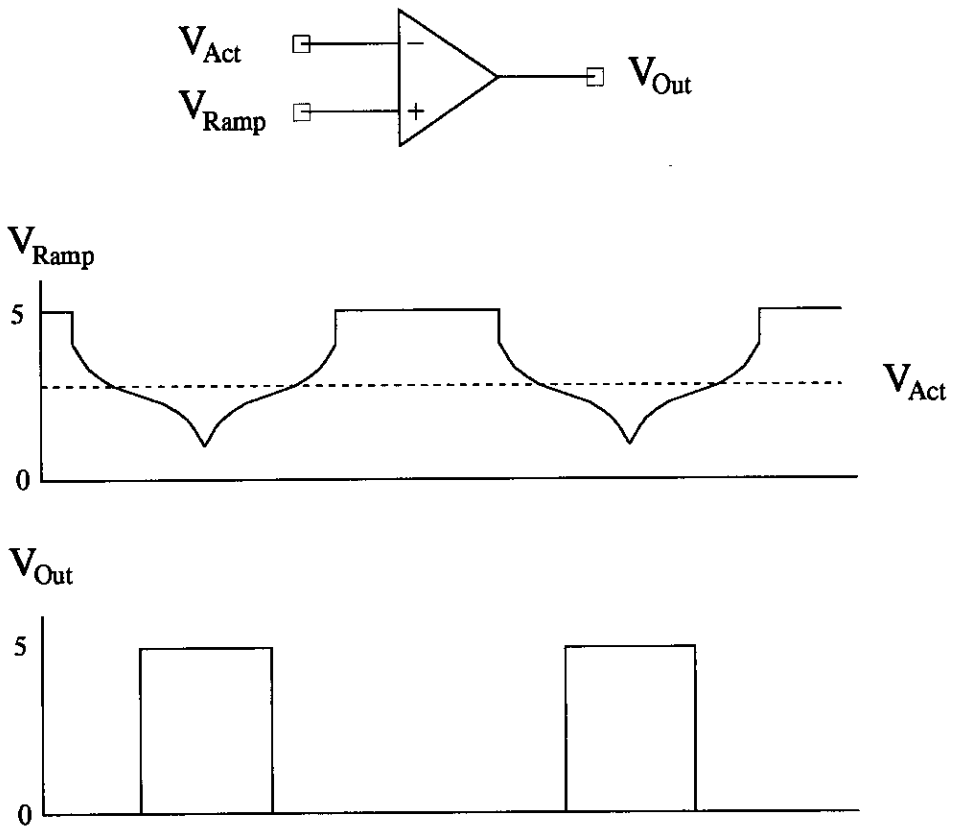


Figure 6.1 Pulse Width Modulation Scheme.

The versatility of EPSILON is due, in part, to its ability to handle either PFM or PWM signals. Thus PFM signals can be used where asynchrony is important, and PWM signals can be used when fast computation is the foremost consideration. Also, to ease the interfacing to analogue voltages, EPSILON can sample and hold 120 analogue voltages internally, converting them into PWM signals via a bank of comparators and a double-sided ramp signal (Figure 6.1). The value of the analogue voltage determines the comparator switch-point, thus controlling the width of the output pulse. The double-sided nature of the ramp ensures that neither the leading nor the trailing edges of pulse width inputs are synchronised. A similar configuration is used to convert the activity voltages into pulse widths. The use of an external ramp to convert an analogue voltage into pulse width allows great flexibility, as the shape of the ramp defines the transfer function of the neuron. So by changing the shape of the ramp the transfer function can be varied from a linear function to a high gain sigmoid. The work concerning the analogue inputs and pulse width outputs was done by Stephen Churcher [117] while the VCOs to convert the activity voltages in PFM signals were designed by Alister Hamilton [118].

Number of Inputs :	120
Number of Neurons :	30
Number of Synapses :	3600 (120x30)
Input Modes :	Pulse Frequency Modulation
	Pulse Width Modulation
	Analogue Voltage
Output Modes :	Pulse Frequency Modulation
	Pulse Width Modulation

Table 6.2 The Specifications for EPSILON 30120PI.

This programmability, cascadability, input/output reconfigurability and process invariance are the main merits of EPSILON.

6.1. Circuit Design Modifications

ES2's 2 μm process (ECDM20) was being phased out as EPSILON was gestating and was only available for prototyping by special order. As a result the ECDM20 process had a high setup cost associated with it. Thus, primarily for funding reasons, EPSILON had to be fabricated using the more economic 1.5μm process (ECPD15/1). This was technically undesirable as the synapse and neuron circuits had been proven using the better toleranced ECDM20 process and the different characteristics of the ECPD15/1 process might create new problems.

Table 6.3 compares the most relevant process parameters for the two different processes for Level 2 typical models. The main differences are summarised below:

1 The transistor β values for the 1.5μm process are about 40% larger for both N-Type and P-Type transistors than the old ES2 2.0μm process. This is due to the field oxide for the transistors now being 40% thinner. Also all transistors are now 40% more capacitive.

2 The threshold voltage for a P-Type transistor is now 0.5V higher.

The above differences in the characteristics of the ECDM20 and ECPD15/1 processes were significant enough to necessitate redesign of the circuits implemented in SADMANN. At the same time the opportunity was taken to incorporate the lessons learnt from SADMANN into the circuits.

Process Parameter	NMOS		PMOS	
	ECDM20	ECPD15/1	ECDM20	ECPD15/1
V_T	0.9V	0.7V	0.6V	1.1V
t_{OX}	40nm	25nm	40nm	25nm
μ_O	$510cm^2/V/s$	$510cm^2/V/s$	$175cm^2/V/s$	$210cm^2/V/s$
N_{SUB}	$0.53 \times 10^{16}cm^{-3}$	$2.0 \times 10^{16}cm^{-3}$	$1.9 \times 10^{16}cm^{-3}$	$5.0 \times 10^{16}cm^{-3}$

Table 6.3 A Comparison of Selected ECDM20 and ECPD15/1 Typical Process Parameters.

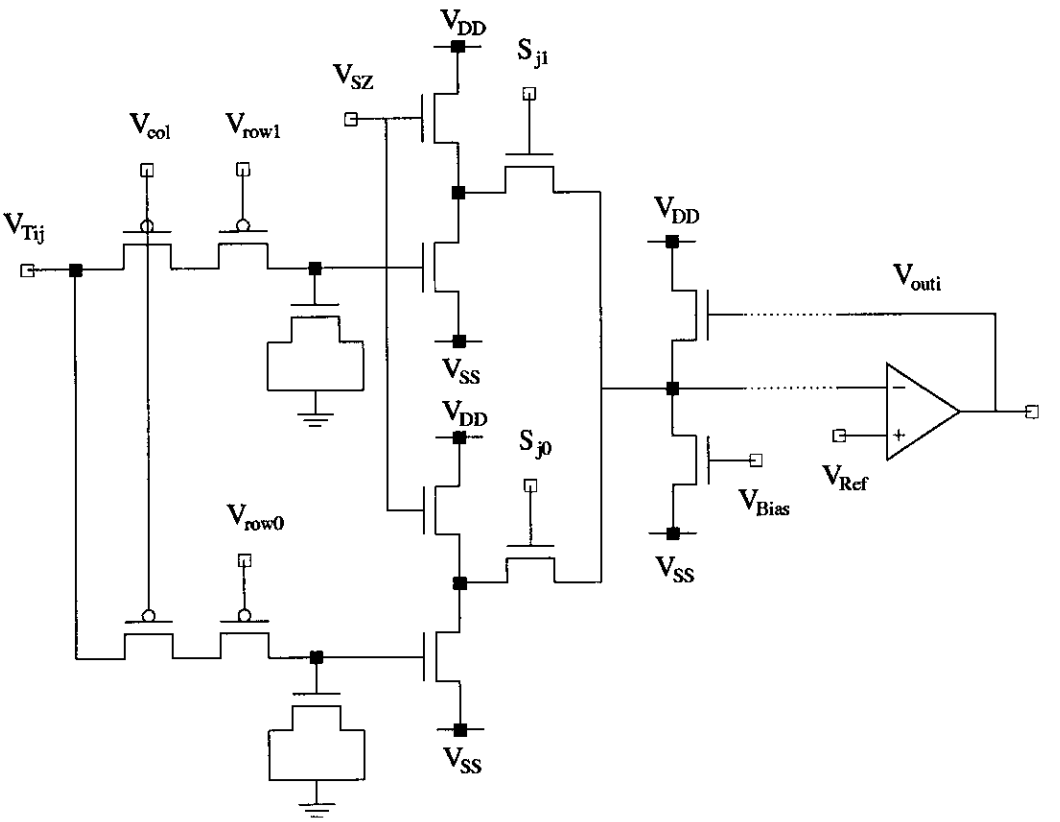


Figure 6.2 The Revised Process Invariant Synapse Design

6.1.1. Revised Synapse Design

The move to the ECPD15/1 process brought about only minor changes to the synapse. The first was to limit the current sourced by the synapse. The transistors in the

transconductance multiplier M1/M2 were simply made longer and thinner to compensate for the "faster" process.

By decreasing the V_{DS} voltages of the transistors in the transconductance stages from 0.8V to 0.5V, the current consumption was reduced further. The new power supply voltages are shown in Table 6.4.

V_{DD}=1.5V	V_{Ref}=1.0V	V_{TijZ}=3.75V
V_{SS}=0.5V	V_{Bias}=3.10V	2.5V<V_{Tij}<5.0V

Table 6.4 New Voltages for the Distributed Feedback Synapse.

The increase in the threshold voltages of the 2 P-type pass transistors for the weight storage capacitor raised the lower limit of the voltage range which they are able to pass. This restricted the usable weight range to 2.5V to 5V, with V_{TijZ}=3.75V.

To minimise the size of the synapse, 2 synapses were implemented as one cell. This reduced the area requirement by simplifying the synapse circuitry to 2 transconductance multiplier stages and just 1 buffer stage (Figure 6.2). Also, area was saved, as the resultant cell has fewer N-wells and thus fewer well crossings in the layout. The size of this "double synapse" was 200μm by 100μm.

6.1.2. Revised Operational Amplifier Design

The tolerances on the ECPD15/1 process are not as tight as those of the ECDM20 process. As a result, the redesign of the operational amplifier was very conservative to guarantee that the amplifier would work. A comparison of the specifications for the ECDM20 and ECPD15/1 versions of ES2's standard cell Operational Amplifier 13 (Table 6.5) illustrates the problems caused by the wider variations of the ECPD15/1 process. Since EPSILON requires at least 30 operational amplifiers, the slew rate and power consumption had to be tightly controlled if the custom operational amplifier was to remain compact and the worst case power consumption contained.

The transistor chain (transistors M8, M9 and M10 in Figure 4.12) which determines the differential stage tail current and output pull-up current, is sensitive to the value of V_T. Small variations in the value of V_T thus have a large effect on the current flowing in the transistor chain. This current not only influences the operational amplifier's slew rate and power consumption but also helps determine the amplifier's gain and phase margin. Thus the ability to determine the value of this current accurately will "tighten" the amplifier specification.

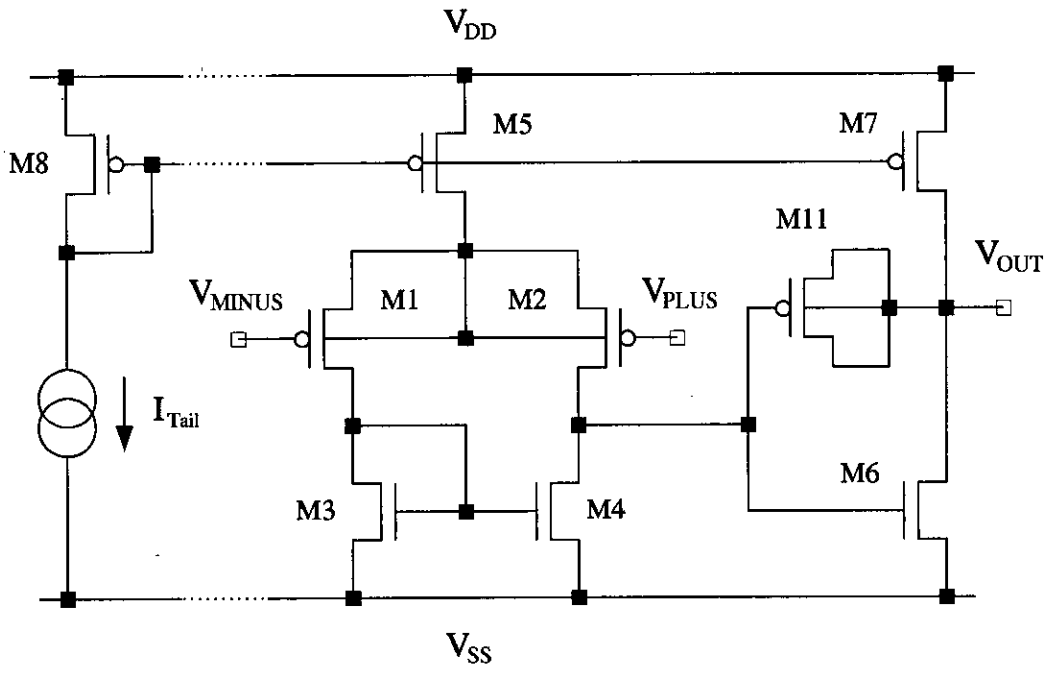


Figure 6.3 A Simple Controllable Tail Current Operational Amplifier Design.

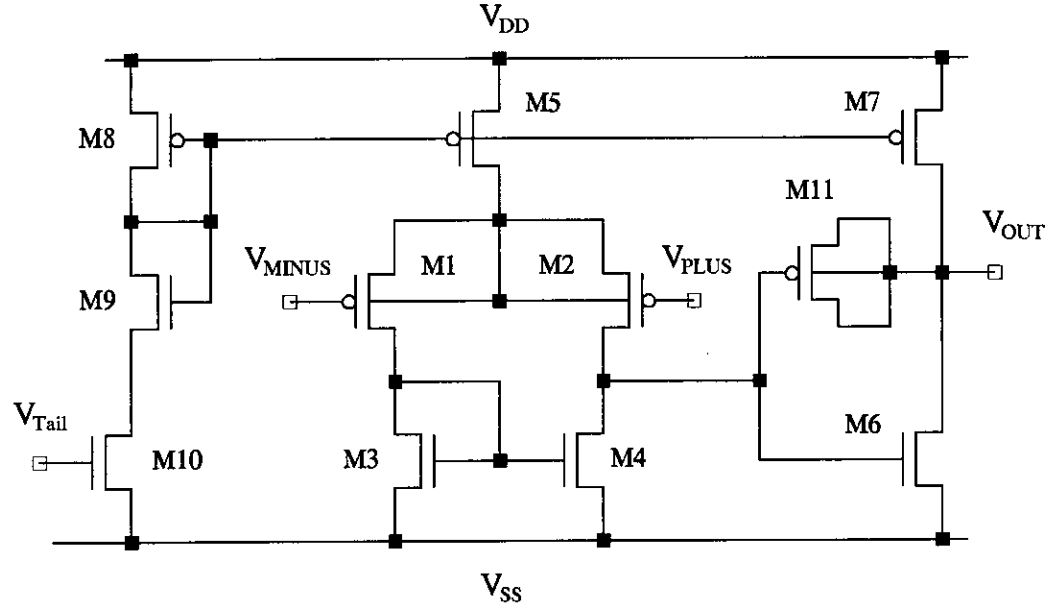


Figure 6.4 The Revised Operational Amplifier Design.

Process	Parameter		Minimum	Typical	Maximum
ECDM20	Slew Rate:	Rise	5.7V/s	18.5V/s	20V/s
		Fall	7.3V/s	18V/s	25V/s
	Power		3.4mW	9.4mW	26mW
ECPD15/1	Slew Rate:	Rise	1.8V/s	7.6V/s	29.6V/s
		Fall	3.4V/s	17.7V/s	47.2V/s
	Power		3.2mW	15mW	56mW

Table 6.5 Slew Rate and Power Specifications for ES2's Operational Amplifier 13 [120, 121].

An obvious approach to determine this current accurately is to replace the transistor chain with an externally controlled current mirror which copies the current to all operational amplifiers within the chip (Figure 6.3). The disadvantage of this arrangement is that fast transients occurring at the inputs or at the output of the operational amplifier will couple onto the "current-set" line of the global current mirror, via the parasitic capacitors in the transistors. This coupling alters the currents in all of the operational amplifiers linked to that current mirror causing their outputs to change.

To decouple the operational amplifiers from each other further, whilst maintaining control over the differential tail current and output pull-up current, the circuits in Figure 6.4 and 6.5 were developed. The chain of capacitors linking the output and input nodes to the "current-set" line of the current mirror is now longer. Thus the coupling onto this line is now smaller due to the greater attenuation effect of the longer capacitor chain.

The "current-set" voltage, V_{Tail}, is determined within the chip and buffered through an external operational amplifier before applying it to the on-board operational amplifiers. The large drive capability of an external operational amplifier enables it to restore V_{Tail} to the correct value quickly. This further improves the stability of the "current-set" line.

HSPICE simulation showed this two-pronged approach of increasing the isolation between operational amplifiers, while turning the common "current-set" line into an external reference voltage, decreased the magnitude of transients on the "current-set" line by a factor of 10.

6.1.3. Revised Voltage Integrator Design

The design changes to the voltage integrator were necessitated by the change of process and the current mirror mis-match problem discussed in Chapter 5.

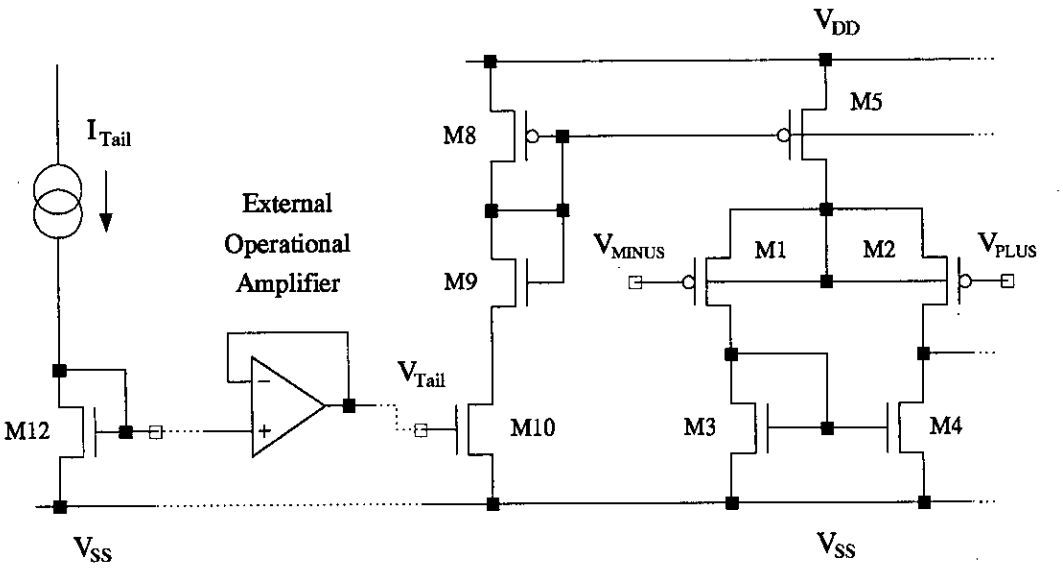


Figure 6.5 The Current Set Circuitry for the Feedback Amplifier.

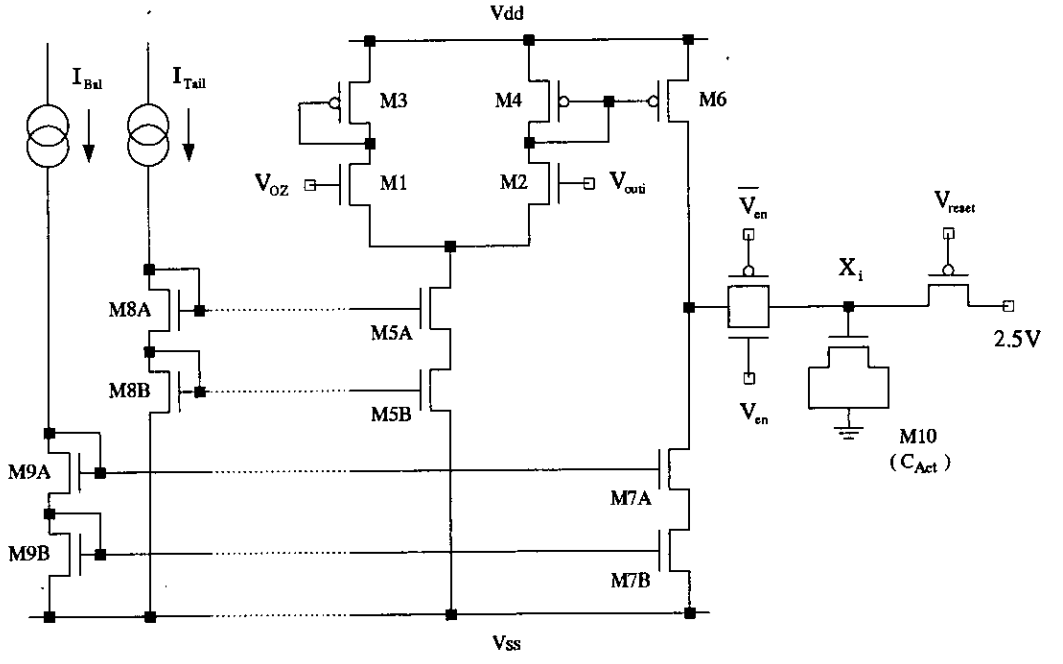


Figure 6.6 The Revised Voltage Integrator Design.

As no definite explanation could be found for the current mirroring problems it was decided that separate current mirrors for the tail and balance currents should be introduced. Independent control of these two current mirrors ensured that the integrator could be balanced correctly if the current mis-match problem in SADMANN recurred.

The much higher P-type threshold voltages meant that there was no longer enough voltage between the power supplies (5V) to allow the P-Type cascode current mirror to operate. This problem was solved by replacing the cascode current mirror with a simple current mirror. To compensate for the resultant loss of mirroring accuracy, the transistors used in this simple current mirror were designed with a long channel length (20μm) to reduce the inaccuracies caused by the lambda ‡ of a transistor.

To ease the interfacing of EPSILON to its support system, two additional features have been included in the voltage integrator circuit.

1 A transmission gate to isolate the output current stage of the integrator from the activity capacitor.

2 A transistor to reset all the activity voltages to a known voltage before the start of computation.

As it is unrealistic to expect the output currents of the voltage integrator to be perfectly matched, the transmission gate is required to prevent the activity voltage from drifting after the synaptic calculation cycle is complete. The ability to reset the activity voltage to a known value is vital for proper multiplication. In its absence all results are relative to different start points, and comparison is impossible.

As a result of these changes there is a significant difference in the way SADMANN and EPSILON operate. In SADMANN as the VCO is always connected to the output of the voltage integrator, the output frequency of the VCO changes smoothly as the activity voltage increases or decreases. The operation of EPSILON comprises of 2 separate phases.

1 Calculate the neural activities.

2 Fire the output neurons.

In between these two phases the output of the synaptic array is disabled. Thus the operation of the synaptic array has been separated from the operation of the output neurons.

The resulting circuit is shown in Figure 6.6.

‡ Lambda is the channel length modulation parameter of a transistor.

Cell	Size
Double Input Neuron	336x100μm^2
Double Synapse	200x100μm^2
Feedback Operational Amplifier	200x262μm^2
Voltage Integrator	200x200μm^2
VCO	200x100μm^2
PWM Comparator	86x116μm^2
EPSILON	10x9.5mm^2

Table 6.6 Cell Sizes for Circuits in EPSILON.

6.1.4. EPSILON Device Details

To complete the system these circuits were combined with the input neurons and PWM output neurons designed by Stephen Churcher [117], and the VCOs designed by Alister Hamilton [118]. Table 6.6 shows the sizes of each component and the die size of EPSILON.

The main data transfer bottleneck in such a large neural VLSI implementation is the loading of the 3600 synaptic weights. To reduce this overhead EPSILON has 2 weight load lines. The estimated worst case power consumption for this device is 350mW which is well within the safe limits (1W) for a chip of this size. Power consumption per synapse is 97μW. The equivalent figure for Intel's ETANN device is 220μW(450mA x 5V / 10240 synapses) [61]. Thus a synapse in EPSILON dissipates under half the power of a synapse in ETANN.

In this implementation the neural state values, $0 \leq V_j \leq 1.0$, are either encoded as PFM signals in the range 0-500kHz (0-50% duty cycle) or as a 0-20μs PWM signal. The analogue input mode is functionally equivalent to the PWM input mode, as the sampled voltages are converted into 0-20μs PWM signals by the input neurons. For both PFM and PWM signals the meaning of the input and output pulse streams is slightly different. For input pulse streams, a pulse width of 0μs and a duty cycle of 0% both represent zero. However the result of reseting the activity capacitor to 2.5V (the mid-point of the PFM/PWM output ranges) is that for the output pulse streams zero activity is represented by a pulse width of 10μs and a duty cycle of 25%. At present this difference is corrected in software.

As the largest package size available contained only 144 pins it was not possible to pin out all of the 120 inputs directly. This restriction resulted in the organisation of the 120 inputs into 4 banks of 30 which are multiplexed in turn onto 30 input pins.

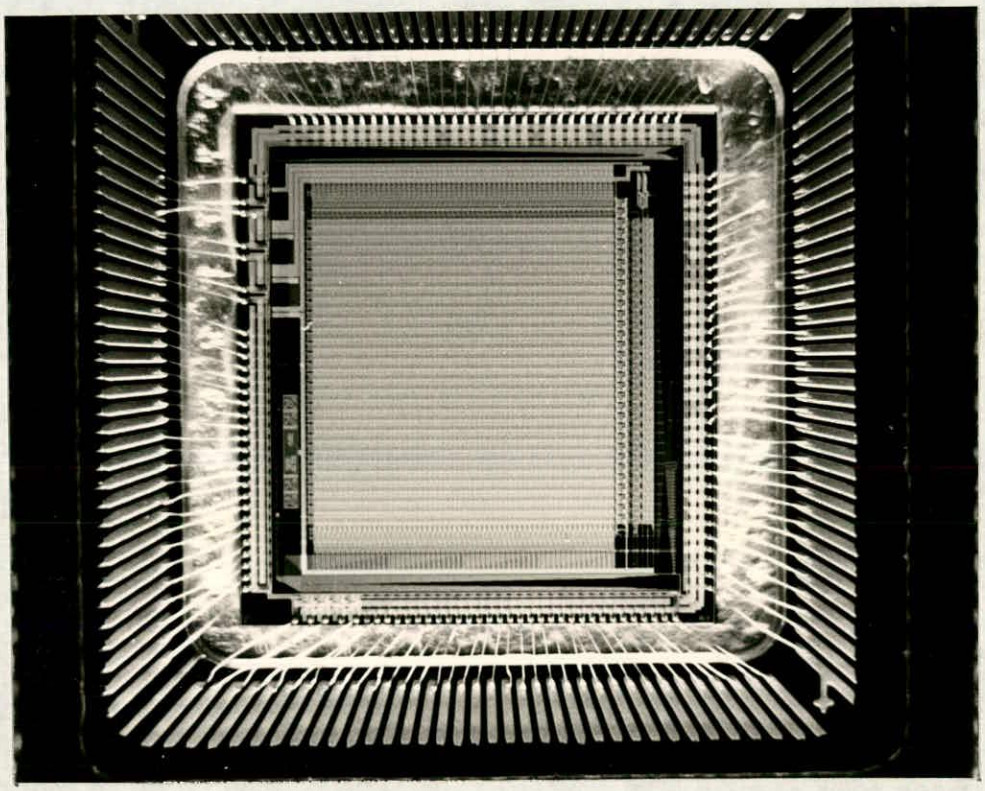


Figure 6.7 Photograph of EPSILON 30120PI.

The operation of EPSILON has 2 distinct phases.

1 The calculation of activity voltages by firing either pulse widths or frequencies into the chip.

2 Evaluation of the neural output states by capturing the output PWM or PFM pulse trains.

Thus, while for PWM signals the 3600 synaptic multiply accumulate operations are performed in 20μs, another 20μs is required to read back the neural state value. 3600 calculations in 20μs gives EPSILON a peak computation rate of 180 MCPS (pipelined).

A photograph of the resultant fabricated device is shown in Figure 6.7

6.2. FENICS (Fast Electronic Neural Integrated Computer System)

The FENICS board to support EPSILON was designed as an autonomous system. A host computer downloads weight and state data to the board and waits for the resultant neural states to be sent back.

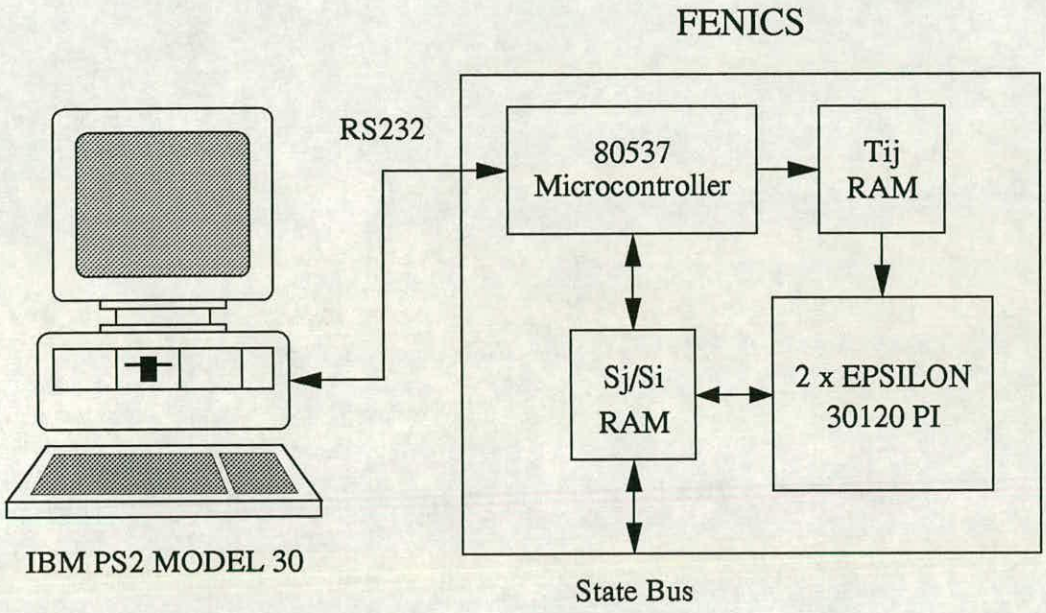


Figure 6.8 A block diagram of FENICS.

As Figure 6.8 shows FENICS contains 4 main function blocks.

1 80537 Microcontroller

2 A state-machine for weight load and refresh.

3 A state-machine for firing the pre-synaptic neural states and capturing the post-synaptic states.

4 Two EPSILON 30120PI integrated circuits.

At the heart of the board is the microcontroller which interfaces to the host computer system via its in-built RS232 interface. The use of the microcontroller allows a degree of intelligence to be built into the board. For example when the microcontroller receives a command followed by data, it automatically performs all the operations necessary to transfer the data to the appropriate state machine. This gives the desired level of autonomy and provides a straightforward user interface.

Once the weights have been loaded into the SRAM of the weight state-machine, the refresh cycle for the capacitively stored weights in EPSILON is transparent to the operation of the FENICS board. During the calculation and evaluation phases of EPSILON, weight refresh is switched off to reduce the effects of noise in the chip, thus enhancing the accuracy of the computation.

The SRAM in the S_j/S_i state-machine contains the bit-streams representing the PFM or PWM input signals. This block also contains circuitry for the generation of the

double-sided ramp used by the comparators in the input and output neurons to convert analogue voltages into PWM signals. The sequence of operations to fire pulses into EPSILON and capture the output pulse signals is as follows.

1 For PFM and PWM input modes the microcontroller enables the clock to the state SRAM address counter. This clocks the pulse signals contained in the state SRAM into EPSILON.

2 In analogue mode, the state values from the microcontroller are loaded into EPSILON via a DAC and two 16-way analogue multiplexers. The double-sided ramp to convert these voltages into PWM signals is generated by clocking the contents of a SRAM containing the ramp data into a DAC.

3 The output PFM signals are captured by enabling the state SRAM write signal while clocking the state SRAM address counter. The 10MHz clock rate of the SRAM address counter resolves pulse signals to an accuracy of $0.1\mu s$.

4 Capture of PWM signals is similar to above, except that while the state SRAM is clocked the ramp SRAM is also clocked. This supplies the output neurons with the double-sided ramp necessary for the production of PWM signals.

State data can be transmitted to and from the microcontroller in 2 ways.

1 Raw SRAM pulse data.

2 Duty cycle, pulse width and analogue voltage (input neural state only).

In the second case the microcontroller has the necessary algorithms built in to convert duty cycles and pulse width data to SRAM data and back again. Unfortunately the microcontroller's $1\mu s$ instruction cycle time is not fast enough to perform the duty cycle and pulse width to SRAM conversion quickly. As a result it is faster to perform the state to SRAM conversion on the IBM PS2 and then download up to 4Kbytes of data to the board.

To facilitate the construction of large networks, such as the 100 by 100 synaptic array required for the proposed neural VLSI optimisation task, the FENICS board has a bidirectional state bus to allow data transfer between multiple FENICS boards.

6.3. Calibration Results

Due to package and die size limitations, only a small number of pads were available for test purposes. As a result, only the output of the feedback operational amplifier and voltage integrator in neuron 0 could be monitored.

The 30 untested EPSILON devices received from ES2 were subjected to a series of 4 tests.

1 Power up test.

2 Clocking data through the x and y shift registers for addressing the synaptic weights.

3 Measuring the V_{SZ} and V_{OZ} values produced by the automatic bias circuits.

4 Quantitative measurement of the multiplication characteristic of neuron 0.

Of the 30, only 3 failed the power up test, with another 11 failing one or more of the remaining 3 tests, leaving 16 working devices. For the large die size (10mm by 9.5mm), a raw yield of just over 50% is very good.

Unfortunately, pressure of time dictated that full calibration of all 16 working EPSILONs was not possible. Thus the following calibration results were based on measurements from only 6 devices. The exception was the automatic bias measurement. As the measurement of V_{SZ} and V_{OZ} was part of the initial test procedure, data was available for all 16 working EPSILONs.

6.3.1. Static Measurements

Automatic Bias Circuits

In accordance with the insight obtained from the implementation of the automatic bias circuits in SADMANN, the feedback loops for generating V_{SZ} and V_{OZ} on-chip contained a column of 64 cells (as opposed to just 1). Also, the link between these circuits and the V_{SZ} and V_{OZ} lines for the synaptic array and voltage integrator is made off chip. This allows the automatically generated biases to be overdriven if their values prove inaccurate.

$$V_{TijZ} = 3.75V \qquad V_{Bias} = 3.10V$$

The above voltages were applied to the V_{SZ} and V_{OZ} bias circuits respectively and the resultant voltages noted. To gauge the accuracy of these voltages, the monitor point on the feedback amplifier in neuron 0 was used to determine the values of V_{SZ} and V_{OZ} manually.

1 V_{SZ} was measured manually by setting $V_{Tij}=V_{TijZ}=3.75V$. Then the "V_{SZ} overdrive" line to the synaptic array was varied until the feedback amplifier's voltage remained constant as all the presynaptic inputs were switched on and off.

2 V_{OZ} was measured manually by simply noting the feedback amplifier's output voltage with all the presynaptic inputs switched off.

Tables 6.7 and 6.8 compare the results obtained from the automatic bias circuits and the manual measurements with the HSPICE Level 2 simulation values.

	V_{SZ}(V)			V_{OZ} (V)		
	Bias Cct	Measured	|Diff|	Bias Cct	Measured	|Diff|
Minimum	4.18	4.10	0.01	3.44	3.43	0.01
Mean	4.252	4.257	0.056	3.621	3.601	0.041
Maximum	4.33	4.34	0.17	3.70	3.66	0.15
Variation	+2.0% -1.7%	+2.0% -3.7%	-	+2.2% -5.0%	+1.6% -4.7%	-

Table 6.7 The Percentage Variations for V_{OZ} and V_{SZ} Bias Voltages for the Bias Circuits and Measured Values. All percentages are expressed relative to the mean responses.

Process	V_{OZ} (V)	V_{SZ} (V)
Level 2 - Slow	4.37	3.72
Level 2 - Typical	4.36	3.71
Level 2 - Fast	4.34	3.70
Variation	+0.2%/-0.5%	±0.3%

Table 6.8 The Simulated Variations in V_{OZ} and V_{SZ} Bias Voltages. All percentages are expressed relative to the typical responses.

There are two main observations:

1 The match between the simulation results and the measured values is reasonable, differing by only 100mV.

2 While the average values for the bias circuit outputs and the measured results are very similar, within the same chip there is an offset of approximately 40-60mV between the two.

The power line resistance problem discussed in the next section means that the power supply voltages degrade from right to left across the array. As the feedback columns and neuron 0 are on the opposite sides of the synaptic array, they will be operating from different power supply voltages. A performance mismatch results.

Synapse failure or transistor defects within the column could also explain this difference. As the areas of the column for the automatic bias circuits and a normal synaptic column are the same, the number of defects they contain

should also be similar. Thus for a large sample size the mean values of the two outputs should be similar. This is in fact the case.

The solution in the short term for EPSILON is to set the V_{SZ} and V_{OZ} manually. Longer term solutions would involve feedback on a more local level, perhaps even per synapse.

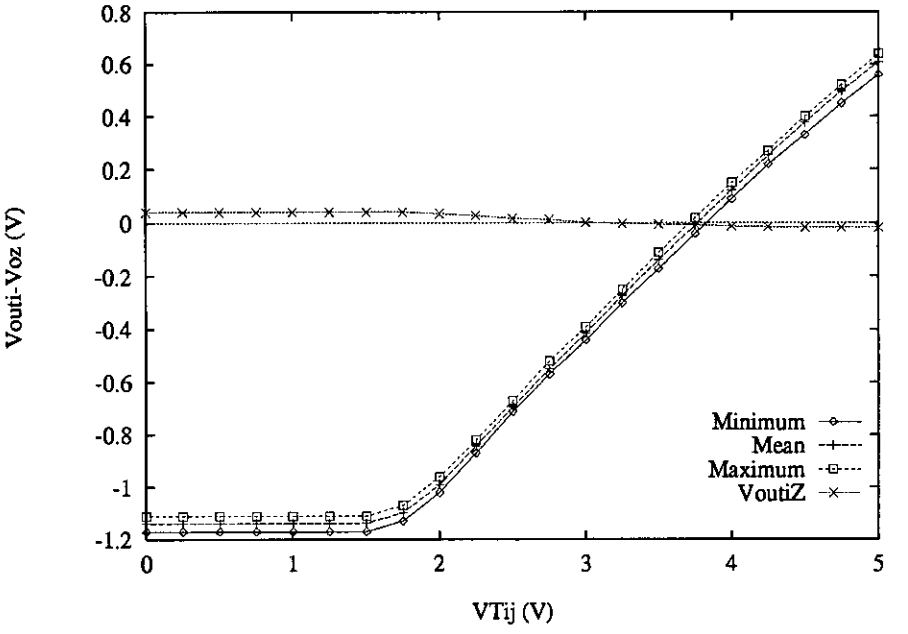


Figure 6.9 Process Variation for Static Weight Sweep.

Static Measurements of Synapse

As the swing in the feedback amplifier output for a single synaptic weight sweep from 0V to 5V is about 15mV (120th of 1.75V), it was impossible to characterise individual synapses. Thus, as for SADMANN, the static synaptic measurements represent a whole column of synapses working in parallel.

As the feedback amplifier output in neuron 0 is the only neuron to have an external monitor point, static synaptic measurements relate to only one synapse column per device. In view of the problems with the automatic bias circuitry described in the previous section, the V_{SZ} voltage reference was set manually for each measurement sweep. All of the presynaptic inputs were switched to 5V (ON) and the whole weight array was loaded with the same voltage.

Figure 6.9 and Table 6.9 show that the worst case variations between the 6 sets of V_{Tij} to $(V_{outi} - V_{OZ})$ characteristics were +5.3% and -7.9%. While this compares

V_{Tij} (V)	$V_{outi} - V_{OZ}$ (V)			
	Min	Mean	Max	Variation
0.00	-1.17	-1.138	-1.11	+4.6%/-5.3%
1.00	-1.17	-1.138	-1.11	+4.6%/-5.3%
2.00	-1.02	-0.990	-0.96	+4.9%/-4.9%
3.00	-0.44	-0.415	-0.39	+4.1%/-4.1%
4.00	0.09	0.123	0.15	+4.4%/-5.4%
5.00	0.56	0.608	0.64	+5.3%/-7.9%

Table 6.9 The Percentage Variations of the V_{Tij} to $V_{outi} - V_{OZ}$ Characteristic. All percentages are taken relative to the mean response for V_{Tij}=5.00V (0.608V).

favourably with the +9.3% and -10.6% worst case variations quoted in Table 5.5 for the equivalent SADMANN results, the limited number of data for EPSILON makes any comparison statistically unsound.

The results confirmed the linearity of the synapse and compare well with HSPICE Level 2 typical simulation values (Appendix 6, Table A6.1). However, the results also revealed a problem. Ideally as the weight voltage is varied from 0 to 5V with the transconductance multiplier switched out, the output voltage of the feedback amplifier, V_{outi}, should remain constant. This was not the case. The output of the amplifier decreases slightly as the weight voltage increases (Trace V_{outiZ}, Figure 6.9).

Since the whole of the synaptic array is set to the same weight voltage, the supply current demanded by the synaptic array changes by a factor of 2 as the weight voltage varies from 0 to 5V. This current variation in turn causes the voltage drop along the power tracks feeding the synapses to change, altering the performance of the synapse. While this voltage drop is probably only 10-20mV, the transconductance multiplier is very sensitive to its power supply voltages.

The resistance of the 200μm by 8μm metal 2 power supply tracks in the double synapse cell is 1Ω. With the transconductance stage of the distributed feedback synapse switched out, the feedback amplifier's voltage is determined solely by the buffer stage. As the weight voltage increases, the current demanded by the transconductance multipliers also increases. This reduces the value of the 1.5V supply for the synapse and increases the 0.5V supply. Thus the V_{DS} and the V_G voltages for transistor M5 (Figure 4.7) in the buffer stage are reduced, while for transistor M4 only V_{DS} is decreased. As the current in transistor M5 decreases more rapidly, the value of V_{outi} falls to compensate.

This is exaggerated by the greater resistance of the 0.5V supply track which has more vias than the 1.5V track. Taking into account vias and the effects of over-etching, we can estimate the cell power track resistance as 2Ω for the 1.5V and 4Ω for the 0.5V line. HSPICE simulations (Appendix 6, Table A6.2) confirmed that these resistances would indeed explain the measured 60mV change in V_{outi} over the 0-5V weight sweep.

As the power supplies enter the array on the right-hand-side of the synaptic array the synapse performance deteriorates progressively right to left across the array. This means that neuron 0 on the left-hand-side of the array, 6.4mm from the power supply bus, experiences the worst effects of the resistance in the power supply lines.

To reduce the power track resistance, the length of tracks should be shortened and their width increased. For EPSILON it would be possible to quarter the track resistance by introducing two power buses, one 1/4 along the array and the other 3/4 along the array. Augmenting this with an increase in the track width within the synapse will reduce the variation in the zero voltage of the feedback amplifier to less than 10mV.

6.3.2. Dynamic Measurements

The initial dynamic tests of EPSILON determined the device's performance under the three possible input modes.

1 Pulse Frequency Modulation

2 Pulse Width Modulation

3 Analogue Voltage.

The extra capacitance caused by having a monitor point on the output of the activity capacitor of neuron 0 gave it a slow response. Thus in calculating the statistics for the dynamic tests in this section, neuron 0 was ignored. For these tests the synaptic column was treated as one synapse with all the weights and input states equal. The output states were measured via the PWM output comparator and a double-sided ramp to give the neuron a linear transfer function. V_{SZ} and V_{OZ} were set manually using the following procedures:

1 To set V_{OZ}, all the presynaptic inputs, S_j, are set to zero and all synaptic weights are set to V_{TijZ}=3.75V. Then V_{OZ} is varied until, after the calculation and evaluation phases of EPSILON, the average of all the output pulse widths, S_i, is 10μs (zero activity).

2 For setting V_{SZ} the average of the S_i values for all S_j off and all S_j fully on are compared ($V_{Tij} = V_{TijZ}$). V_{SZ} is then varied until the means match.

Appendix 7 contains the graphs of the mean response of neurons 1 to 29 in EPSILON chip 3, to a double sweep of input state values and weight voltages, for the

above 3 input modes. These graphs demonstrate that for all 3 input modes the synaptic multiplication characteristic is indeed linear.

The voltage offset problem which was present in the voltage integrator in the SADMANN device is not present in EPSILON. This reinforces the suspicion that, since the two circuits were very similar, the problem in SADMANN was indeed a layout problem rather than a circuit problem.

During this series of tests it became clear that there was a peculiarity during switching associated with the feedback operational amplifier. The multiplication characteristic was as expected for analogue and PWM input modes but for PFM signals the multiplier characteristic was non-linear. This was due to an oversight in the routing of signals from the core of EPSILON to the pad ring. The state input signals come directly from the pads on the left-hand-side of the chip to the left-hand-side of the synaptic area. The tracks for the current mirror lines for the feedback operational amplifier start at the top left-hand corner of the die and enter the core of the chip at the bottom left-hand-side of the synaptic array, passing as they go under all 30 of the state input lines. The coupling from these input lines thus has a significant effect on the current set line of the feedback amplifiers. The greater the number of pulses (edges) on the input lines the larger the effect on the feedback amplifier's output voltages. This explains the difference in performance between the PFM input mode and PWM/Analogue input modes. For this test, due to all the inputs working in parallel, all edges were synchronised, giving a large "kick" to the operational amplifier every time an edge occurs. By introducing a random offset between the input pulse streams, the edges of the PFM signals become asynchronous, reducing the effect on the operational amplifier current set line. The PFM characteristic in Figure A7.1 was measured in the presence of just such a random offset.

The next stage was to look at the performance of individual synapses. Measurement of a single synapse was, however, impossible as its small response at the feedback operational amplifier, 15mV, would be overwhelmed by noise. The 120 by 30 synaptic array in EPSILON was therefore configured as a 10 by 30 array for this series of tests.

PWM input signals were chosen for two reasons.

1 The large number of edges present in PFM signals create unwanted noise due to the coupling onto the current set line of the feedback amplifier.

2 The FENICS hardware allowed the pulse width range to be extended to 0-100μs, so that the response due to the groups of 12 synapses was amplified, making better use of the dynamic range of the output measurement system (0-20μs in 0.1μs steps). N.B. The hardware limited the PWM signal generated inside EPSILON from analogue inputs from 0 to 25μs.

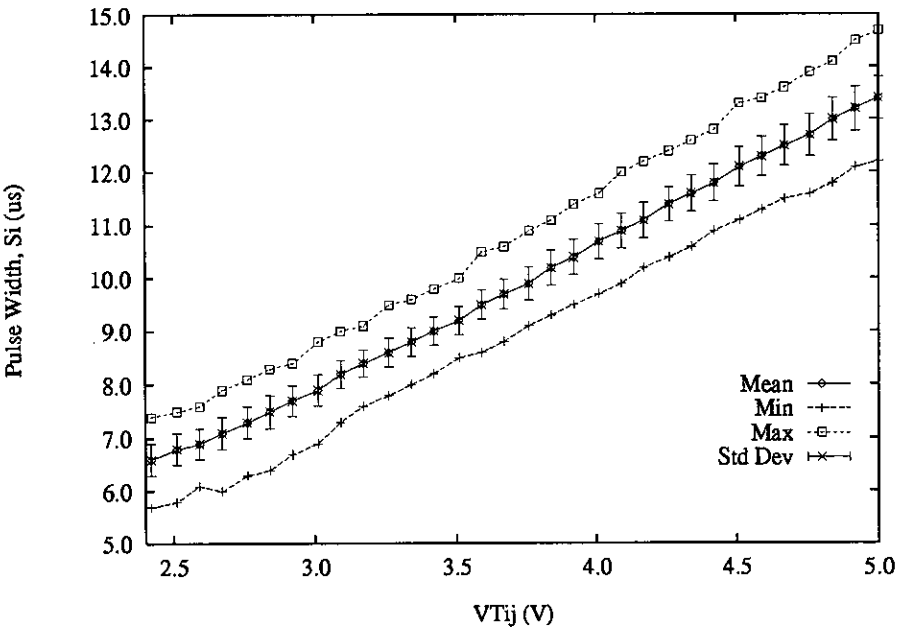


Figure 6.10 Process Variation for Weight (V_{Tij}) Sweep.

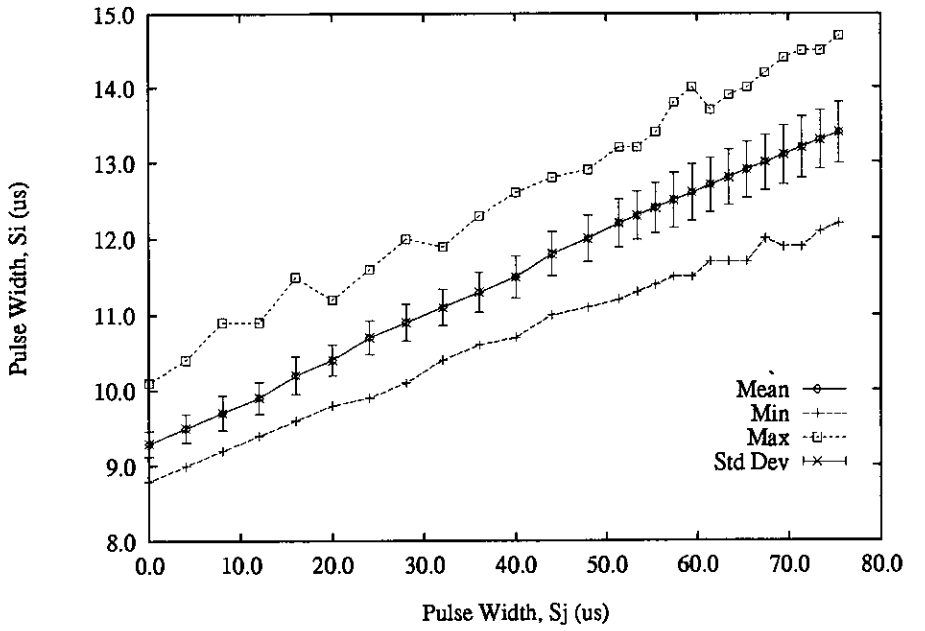


Figure 6.11 Process Variation for Pulse Width (S_j) Sweep.

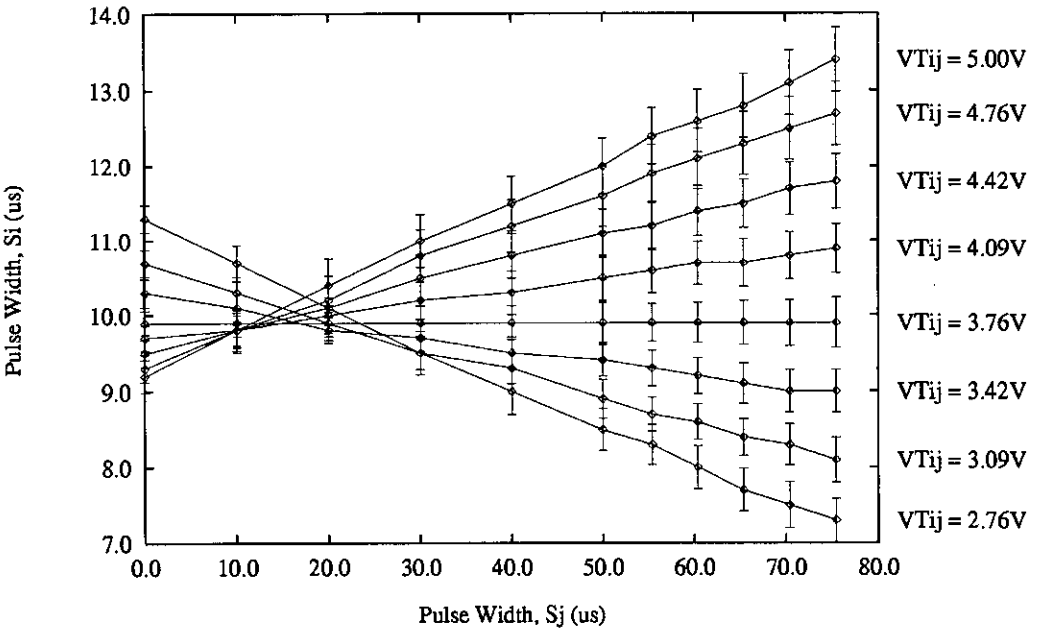


Figure 6.12 Process Variation for Double Sweep.

Initial attempts to characterise the groups of 12 synapses proved problematic, thanks to imbalances within the voltage integrator created by transistor mis-matches. These caused the integration capacitor voltage to drift. It should be pointed out that these mis-matches are simply due to the effects of process variations between transistors in the same current mirror rather than the systematic offset problem which occurred in SAD-MANN. The voltage drift created by this current mirror mis-match was eliminated by using one of the groupings of 12 synapses within each synaptic column as a bias to compensate for the drift. A software routine on the IBM PC host computer altered a bias value iteratively until either the drift had been cancelled or the bias value reached the end limits of its influence. If a particular synaptic column could not be correctly biased then the results from it were ignored in calculating the statistics.

It should to be pointed out that the use of extended pulse widths and characterising synapses in groups of 12, implies that the process variation figures below are only an indication of the variance in synaptic multiplications rather than a precise value. For comparison with the results from SADMANN the same 3 sweeps were carried out.

	Sweep	Range	Step Size	
1.	V_{Tij} sweep	2.5-5.0V	0.08V	(S_j=100μs)

2.	S_j sweep	0-100μs	4μs	(V_{Tij}=5.0V)
3.	Double sweep	0-100μs 2.5-5.0V	10μs 0.32V	

To eliminate noise from the measurements and increase accuracy, 10 samples were taken for each measurement and the average used as the system response for that measurement.

Sweep	S_j (μs)	V_{Tij} (V)	S_i (μs)				
			Min	Mean	Max	Variation	Std Dev
V_{Tij}	75.4	4.90	12.10	13.20	14.50	+38.2%/-32.4%	±12.3%
S_j	75.4	5.00	12.20	13.40	14.70	+38.2%/-35.3%	±12.0%
Double	75.4	5.00	12.10	13.40	14.50	+32.4%/-38.2%	±12.2%

Table 6.10 Worst Case Process Variations for Dynamic Measurements of S_i. All percentage are expressed relative to the mean response for V_{Tij}=5.0V and S_j=100μs (13.4-10.0μs)

Figures 6.10, 6.11 and 6.12 contain the responses to the 3 sweeps and Table 6.10 summarises the worst case process variations in each sweep.

Resetting the integrator capacitor to 2.5V at the start of each calculation/evaluation cycle means that 10.0μs represents the zero activity value of the system. Thus, all S_i values have to be referenced to 10.0μs. For example, in the calculation of percentages, the system response used as the reference value was 3.4μs and not 13.4μs (V_{Tij}=5.0V and S_j=100μs).

These results again confirm the linearity of the distributed feedback synapse at the heart of EPSILON. The other salient points are as follows:

1 As the scales in Figures 6.11 and 6.12 show, rather than generating 0-100μs pulse widths, the pulse outputs from the S_j/S_i SRAM on FENICS were 0-75μs. Pulse widths in the 0-50μs range were output from the RAM without any problems, but 50-100μs pulses were compressed into 50-75μs pulses. The problem was thought to lie with the incorrect loading of the count sequence length into the counter which clocked pulse signals out of the state SRAM. During normal PWM operation (0-20μs pulses) there is no such compression.

2 The double sweep in Figure 6.12 clearly shows how the power supply line resistance distorts the position of the zero crossing point of the multiplier characteristic. Since

only 10% of the synaptic column is active during these sweeps, the effects of the offset caused by the other 90% of the column are of the same order. When the whole column is active (Figure A7.3) the significance of the problems caused by the power supply line resistance is greatly reduced.

3 The process variation percentages in Table 6.10 are a factor of 2 greater than those for the SADMANN device (Table 5.7). The primary reason for this increase in process variance is that the tolerances on the ECDM20 process used for SADMANN are much tighter than those for the ECPD15/1 process used for EPSILON. As a result the matching between transistors in the synapse and in current mirrors is not as good, resulting in greater performance variations. Another contributory factor is that 1 or 2 synapses within the groups of 12 tested could contain processing defects. This would distort the overall performance of that synaptic grouping.

To estimate the reliability of the neural calculations, the weight and pulse width experiments were re-repeated for chip 3, row 1, column 0, with 50 samples taken for each combination of V_{Tij} and S_j. Tables A9.1 and A9.2 in Appendix 9 show the worst case variations and standard deviations. Although in both cases the typical worst case variation and standard deviation were ±15% and ±8% respectively, there were several measurements in which the "rogue" samples resulted in standard deviations of ±20-25%. Expressing the sample variance figures as pulse width variations, $\pm 0.51\mu s$ and $\pm 0.27\mu s$ respectively, and comparing them with the $0\text{-}20\mu s$ output range, and the $0.1\mu s$ resolution of the pulse width output mode, puts the sample variance figures into a better context. Furthermore, the characterisation procedure drives EPSILON in a non-standard environment. As a result, these measurements are not representative of normal operation. Even so, results for the same weight and state data could differ by up to $\pm 0.5\mu s$ between runs.

In conclusion, the degree of transistor matching required for the performance of the circuits to be process invariant is not available within the ECPD15/1 process. Fundamentally the problem is that analogue circuits are being fabricated on a digital process (ECPD15 = ES2 CMOS Philips Digital $1.5\mu m$ process). This means that even current mirrors cannot be depended upon to copy current accurately from one part of a circuit to another. Feedback on a more local (ideally synaptic) level would solve the problem of the non-uniformity of the ECPD15/1 process. The problems caused by the resistance of the synapse power supply lines simply reinforces the knowledge that large scale analogue neural systems require very careful design.

Despite the limitations of the process, the circuits perform very well, combining to form a system which has a linear response and has a higher level of process invariance than the synapse fabricated by Hamilton [118] despite being produced on a more variant process.

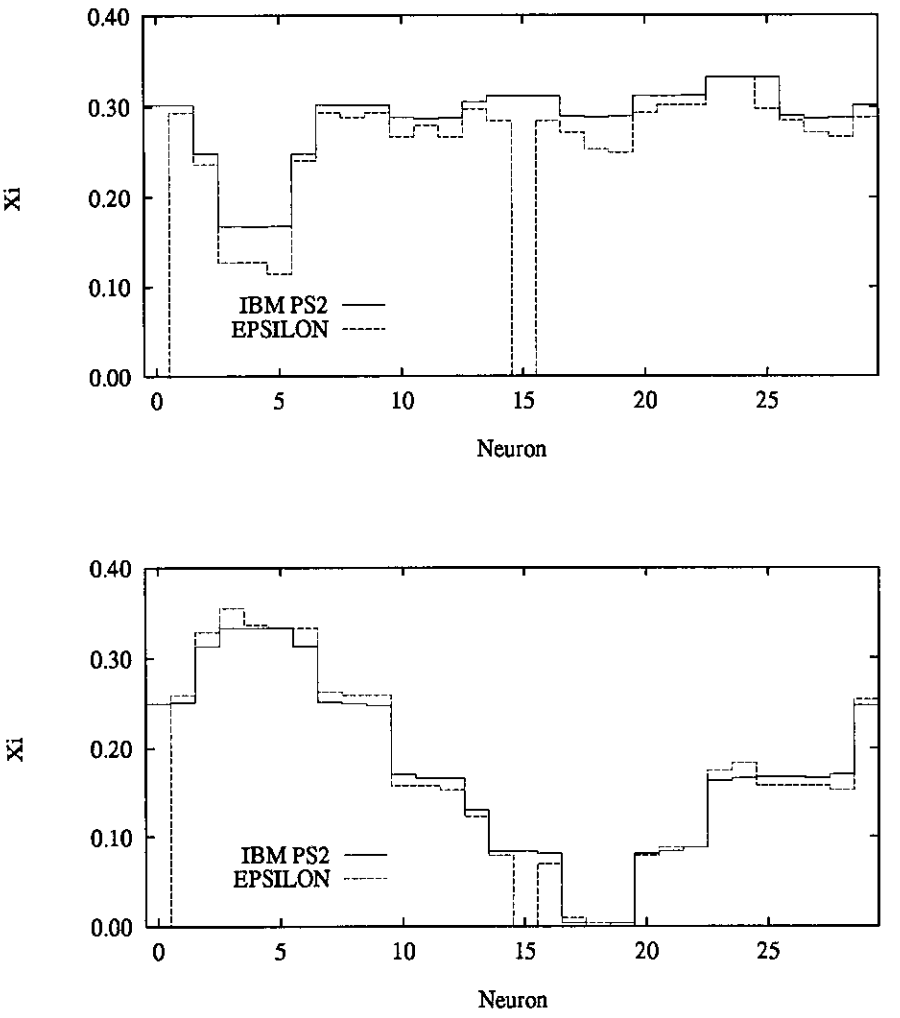


Figure 6.13 Software and Hardware Responses for the 9 city data set.

6.4. Kohonen Demonstrator

Unfortunately, pressure of time prevented the completion of the full 2 board, 4 chip neural optimisation system. However, to determine the feasibility of the analogue VLSI hardware for solving optimisation problems, a small-scale problem was tested on the hardware available.

At that time the hardware consisted of 1 FENICS board incorporating a single EPSILON device. The limited number of neurons available dictated that the test problem should be the 10 city TSP solved via the Kohonen Self-organising neural network.

EPSILON contains only synaptic multipliers and neurons with linear or non-linear transfer functions. There is no built in mechanism for adjusting the values of synaptic

weights. Thus the weight update calculations for the Kohonen network were performed on the host computer. For this test problem, EPSILON's PWM input and output modes were used.

The software/hardware comparison was performed as follows for each TSP data set.

1 Solve TSP using floating point precision on the IBM PS2 host computer.

2 Download final weight set to FENICS board.

3 In turn present each city vector to EPSILON and measure the neurons' output response. Determine best match neuron in each case, and calculate resultant tour length.

4 Calculate the ideal response to the city vectors in software.

The first comparison was performed using the Hopfield/Tank 10 city TSP example. As Table 6.11 shows EPSILON did not produce the same tour as the software simulation. EPSILON grouped cities 0, 1 and 2 incorrectly onto neuron 6, and cities 8 and 9 onto neuron 13. This confusion was attributable to the mis-matches in the synaptic multiplications masking out the small differences in response between neurons associated with cities close together. For this particular city set, cities 8 and 9 are extremely close together, requiring very high accuracy to separate the responses of the neurons associated with the cities correctly. Thus this data set is, perhaps, a difficult one. As Table 6.11 illustrates, however, EPSILON's responses to the 10 city vectors were a good attempt at a hard problem.

To provide more information about EPSILON's ability to discriminate between the the responses of neurons, the Kohonen neural network was applied to 3 progressively tougher artificial city sets.

1 **4 City Data Set:** The city coordinates form the vertices of a unit square {(0,0),(0,1.0),(1.0,0),(1.0,1.0)}.

2 **5 City Data Set:** Similar to 4 city data set but an extra city has been added at the centre of the square {(0,0),(0,1.0),(0.5,0.5),(1.0,0),(1.0,1.0)}.

3 **9 City Data Set:** A 3 by 3 grid of cities {(0,0),(0,0.5),(0,1.0), (0.5,0),(0.5,0.5),(0.5,1.0), (1.0,0),(1.0,0.5),(1.0,1.0)}.

As can be seen from the above data sets, the cities have been positioned as far apart as possible to increase the separation between neurons' responses. This was to provide EPSILON with a better chance of deciphering the positions of cities in the tour correctly. For all three of these data sets EPSILON identified the same tour for the software weight set as the software simulation.

City	Coordinates	Best Match Neuron	
		Software	EPSILON
0	0.4000,0.4439	5	6
1	0.2439,0.1463	10	6
2	0.1707,0.2293	8	6
3	0.2293,0.7610	2	1
4	0.5171,0.9414	28	26
5	0.8732,0.6536	24	17
6	0.6878,0.5219	21	21
7	0.8488,0.3609	18	17
8	0.6683,0.2536	15	13
9	0.6195,0.2634	13	13

Table 6.11 A Comparison of the Software and EPSILON Best Match Neurons for the Hopfield/Tank 10 city TSP Data Set.

The two graphs in Figure 6.13 compare the ideal (software) response of the 9 city TSP weight set and the vectors for cities 5 and 9, with the response of EPSILON for the same weight and state data. The responses from neurons 0 and 15 of EPSILON were not used. The activation capacitor 0 has extra capacitance due to the external test point connected to it. Neuron 15 was ruled out by the autobias procedure.

To map the output pulse width range on to the 0-0.333 software output range, the mean of EPSILON responses for zero activity and maximum Kohonen activation values were calculated. This information was then used to convert the outputs from EPSILON into values which could be compared directly with the software outputs. Appendix 10 details the variations between the EPSILON and software responses for cities 5 and 9.

For the city 5 vector the match between the two responses varies from -16.2% to 0%, whereas for the city 9 vector the match varies between -5.4% and +6.6%. This does not compare well with the 1.2% deviation between software and hardware results, reported for the Brownlow switched-capacitor synapse [107]. There are four reasons for this significant difference in multiplication accuracy.

1 The switched-capacitor synapse was fabricated on the more uniform ECDM20 process. The results from SADMANN and EPSILON demonstrate that the ECPD15/1 process is twice as variant as the ECDM20 process.

2 EPSILON's uniformity is degraded by the voltage gradient in the synapse power supplies caused by track resistance.

3 The multiplication accuracy of the Brownlow implementation is for a column of synapses whereas the accuracy figures for EPSILON are for a whole device [122]. Thus the figures for the Brownlow implementation do not include the variations in performance of the integrators at the end of each synaptic column.

4 EPSILON is a complete system comprising synapse, integrator and linear/non-linear transfer function. The Brownlow implementation comprises only synapses and integrators. Thus the figures for the accuracy of EPSILON are for a complete system while the figures for the Brownlow synapse are only for a partial system. As there are fewer components in the Brownlow implementation, the cumulative effects of mis-matches between transistors are less.

Taking the above four factors into account, the performance of the Brownlow circuits and EPSILON are probably equivalent, but the differences between the implementations are such that any direct comparison is difficult. The distributed feedback synapse in EPSILON has two main advantages over the Brownlow synapse.

1 Cascadability

2 Flexibility: both PFM and PWM signals can be used.

Thus overall, the circuitry in EPSILON offers the better combination of process invariance, linearity, flexibility and cascadability.

The non-uniformity of the fabrication process within a die is the main factor limiting EPSILON's accuracy. The problems with controlling the effects of process variations have already been discussed at length in the previous section, with the conclusion that without resorting to process compensation circuitry within the synapse (large area penalty) the uniformity of multiplication cannot be significantly improved for this process.

Another factor detrimental to the performance of EPSILON in this context, is the fact that the Kohonen network uses only a fraction of the available output dynamic range. The distribution of weight and state data in networks is such the occurrence of all state inputs and weights fully on is very rare. For example in the Kohonen network for the TSP the magnitude of the input vector is always 1, while the magnitude of the vector with all 3 inputs to the Kohonen network fully on (1.0) is 3. Thus only a third of the available dynamic range is used. The consequence of this limited use of the dynamic range is that EPSILON is required to distinguish accurately between neuron outputs differing by less than $1\mu s$. Thus the use of the Kohonen network is a very tough test of an analogue VLSI system. Bearing in mind the levels of process and measurement variation reported in the last section, EPSILON is performing remarkably well in being able to correctly determine the tours for the 4, 5 and 9 city data sets.

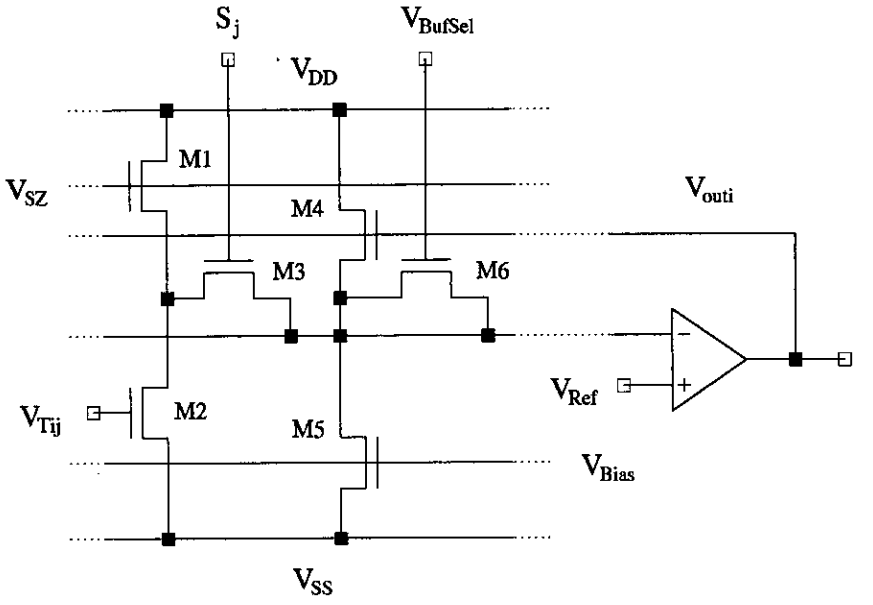


Figure 6.14 Distributed Feedback Synapse with Switchable Buffer Stage.

To exploit the full dynamic range of a neuron's output, the gain between the synapses and the voltage integrator should be variable. This can be achieved in one of three ways in the present version of EPSILON.

1 Increasing the reference currents for the voltage integrator.

2 Extending the length of the input pulse widths.

3 Changing the slope of the double-side ramp to make it more sensitive to the activity voltage.

In all cases the amplification of the synaptic sum is taking place after the feedback operational amplifier has compressed the sum of the synaptic multiplications into its 1.2V output range. Consequently, the above 3 techniques amplify both the synaptic sum and noise component of the signal. As the double-sided ramp signal is calculated in software on the IBM PS2 and then downloaded to the FENICS, the slope of the ramp can be easily and quickly altered. For the present version of EPSILON, this is the most appropriate technique for increasing the exploitation of the available dynamic range.

To avoid amplifying the noise at the feedback amplifier's output, the amplification of the synaptic sum has to take place before the sum is compressed by the feedback amplifier into the ±0.6V voltage range which drives the voltage integrator. This can be achieved by adding transistor M6 to the synapse circuit (Figure 6.14) to make the buffer stage switchable. Analysing a column of these circuits in Figure 6.14 in a similar manner

to the analysis of the distributed feedback synapse in Section 4.2.2 yields:

$$V_{outi} = \frac{1}{M}\frac{\beta_{Trans}}{\beta_{Buf}}\sum_{j=0}^{N-1}(V_{Tij} - (V_{SZ} - V_{Ref} - \Delta V_T)) + V_{Bias} + V_{Ref} + \Delta V_T \quad (6.1)$$

where N is the number of synapses within the synaptic column and M is the number of buffer stages selected within the column. Decreasing the number of buffer stages switched in within a synaptic column, increases the sensitivity of the feedback amplifier's output to the value of the weight voltages. Thus the value of the synaptic sum can be amplified in a controllable manner. By enabling just one buffer stage per column, the effects of a single synapse can be measured accurately. This greatly improves the testability of a large scale implementation.

6.5. Conclusions

Unfortunately, while EPSILON is performing very well, it and probably most other analogue neural VLSI implementations on digital processes do not have the raw accuracy required to run the Kohonen neural network. Subsequent work by Hamilton [118] uses EPSILON to run a MLP network for vowel recognition. The Virtual Targets training algorithm [123] used to train the network successfully compensated for synaptic mismatches in a way not possible with the networks described here.

The effects of noise within EPSILON and in the supporting system FENICS, meant that the results of synaptic multiplications were only repeatable to within $\pm 0.3\mu s$. Thus accuracy of the system is not only limited by the harmful effects of process variations but also by the noise level within the system. For the Hopfield/Tank and the Kohonen networks the effects of noise could prevent the networks from converging to a valid solution. In a MLP network, noise extends the training time of a network and limits the network's classification abilities. Only careful integrated circuit and PCB design can reduce the noise levels in the system, and the associated uncertainty in the results of the synaptic calculations.

The important conclusion is that unless the neural network actively compensates for process variations, the performance of analogue neural VLSI hardware fabricated on digital processes will be distinctly non-optimal. As a result, networks based on learning algorithms such as the delta rule, back propagation and virtual targets will perform well on analogue VLSI hardware providing the VLSI implementation is included in the learning phase of the network. The Hopfield/Tank and the Kohonen networks which depend on the synaptic calculations being uniform are not suited to analogue VLSI circuits implemented on digital processes. Access to a well setup analogue process would improve matters, probably reducing synaptic variations to about 1%. Since the Kohonen network requires very high multiplication accuracy to be able to discriminate correctly between

neurons, even a synapse uniformity of 1% is probably not good enough. For such a process the performance of analogue circuits is limited by the effects of noise rather than the effects of process variations. Thus the noise levels within the system are just as important as the process tolerances in determining the performance of an analogue VLSI neural implementation.

Chapter 7

Conclusions and Discussion

This chapter brings together the results of the software simulations, the hardware characterisation measurements and the Kohonen neural network implemented on analogue VLSI hardware. Conclusions are drawn both about the performance of the Hopfield/Tank and the Kohonen neural networks as optimisation techniques, and the process invariance of the fabricated analogue VLSI hardware. Recommendations are then made about future improvements to the VLSI circuitry.

7.1. Neural Optimisation Network Simulations

The software simulations of the Hopfield/Tank and the Kohonen neural networks confirmed their abilities to produce optimal solutions to the Travelling Salesman Problem (TSP). The 3-way comparison between the performance of the above two networks and an exhaustive technique, applied to 100 randomly generated 10 city TSP data sets, proved very revealing:

1 The Kohonen network found the optimal tour for 73 of the city sets and in the remaining 27 cases the length of the tour was always better than the average of all possible valid tours for that city set.

2 The Hopfield/Tank network found the optimal tour 65 times, in 3 cases was not able to find any tour, while the remaining 32 tours were shorter than the average tour length.

Thus in terms of tour quality the Kohonen neural network performs significantly better than the Hopfield/Tank network. The failure of the Hopfield/Tank network to find any tour in 3 of the city sets illustrates the network's sensitivity to the energy function, which is partly derived from the coordinates of the cities. Importantly, when a tour was produced by either network it was always better than the mean of all possible valid tours. Thus while the networks did not always find the optimal tour they did in general produce an acceptable tour.

The Kohonen neural network with a run time of 3s (Sun IPC Sparc Workstation) for the 10 city TSP was by far the fastest of the 3 techniques. The Hopfield/Tank network required typically 24s for each of the 100 runs of the network while the exhaustive search took 75s to find all possible tours.

As the 10-city TSP was a relatively small optimisation problem it was important to examine how the performance of the three optimisation techniques changed as the problem size increased. Once again the Kohonen network was the clear winner. It found acceptable tours for problem sizes as large as 1000 cities and its computational load scales as the square of the number of cities. An exhaustive search will always find the optimum tour. However, for TSP sizes greater than 10 cities the compute time becomes prohibitive, as the computational load increases exponentially with the problem size. For example, an exhaustive search solution to the 50 city TSP would take 10^{51} years, based on the timings for the 10 city TSP. Unfortunately as the problem size is scaled up the sensitivity of the Hopfield/Tank network to its parameters is heightened. As a result persuading the network to converge to any solution becomes a major problem.

The superior combination of speed, scalability and high quality tours of the Kohonen neural network gives it the clear edge in this 3 way comparison. However, because the Kohonen network and the TSP are very well matched, the Kohonen network may perform very differently on task assignment and scheduling problems.

7.2. SADMANN and EPSILON

The performance of the synapse and neuron circuits, designed to improve the uniformity of synaptic multiplications, was initially proven in a testchip (SADMANN) before the large scale, generic neural building block device (EPSILON) was fabricated. This provided two sets of measurements for the distributed feedback synapse and voltage integrator.

1 As expected the overall multiplication characteristic for the distributed feedback synapse and the voltage integrator was highly linear.

2 The static measurements of the synapse in both SADMANN and EPSILON correspond very well with the results of HSPICE Level 2 simulations. Thus any future design work based on this synapse can be simulated in HSPICE with a high degree of confidence.

3 The voltage integrator in the SADMANN testchip had an offset voltage due to a systematic mis-match between different legs of the same current mirror (M4/M6 in Figure 4.15). No definite explanation for this offset could be found. However, a very similar version of the voltage integrator worked without any problems in EPSILON. The problem in SADMANN was thus thought to be attributable to non-identical layouts around the transistors in the different legs of the current mirror. It appears that this created the systematic mis-match between transistors.

4 The automatic bias circuits produced the desired reference voltages but the values proved unrepresentative of the average response of the synaptic array. This, in

SADMANN, was due to the positioning of just one feedback cell at a corner of the array and in EPSILON to defective cells within the feedback column. To increase the match between the performance of feedback circuits and the synapses, there should have been a column of feedback cells positioned in the middle of the synaptic array.

5 The combination of the long power tracks and the 20-40mA synaptic current consumption resulted in a voltage gradient in the power supply lines across the synaptic array. While this voltage drop was probably only 10-20mV, the effect on the synapse was significant due to its sensitivity to the value of its power supplies. Multiple power buses within the synaptic array to shorten track lengths and wider power tracks would alleviate this problem.

6 An oversight in the routing between the synaptic array and the pad ring resulted in the current set line for the feedback operational amplifier passing directly under all 30 pulse input lines. The coupling effects from the large number of edges in a PFM signal resulted in a significant distortion of a synapse's multiplication characteristic. Better routing would have prevented this problem.

Device	Process	Percentage Variation (Standard Deviation)
Hamilton Synapse	ECDM20 - 2.0μm	±18.1%
Cascode Current Mirror (M8/M7 - Figure 4.15)	ECDM20 - 2.0μm	±3.2%
SADMANN	ECDM20 - 2.0μm	±6.1%
EPSILON	ECPD15/1 -1.5μm	±12.4%

Table 7.1 The Percentage Variation in the Performance of the Hamilton Synapse, a Cascode Current Mirror, SADMANN and EPSILON.

7 Since the ECDM20 process became unavailable for prototyping shortly after the fabrication of SADMANN, EPSILON was fabricated using ES2's ECPD15/1 process. The disadvantage of the smaller geometry was that the tolerances of the process were greater than the ECDM20 process; thus with the ECPD15/1 process there was greater potential for the mis-matching of transistors. As both the distributed feedback synapse and the voltage integrator depend on the process within a single die remaining approximately constant, any mis-matches between transistors degrades the uniformity of the synaptic multiplications. Table 7.1 shows that the wider tolerances of the ECPD15/1 process did result in a decrease in the consistency

of the synaptic multiplications.

In SADMANN the standard deviations of the static process variation measurements for the distributed feedback synapse and the voltage integrator were ±3.4% and ±7.8%, respectively. Thus the overall ±6.1% variation is less than the sum of the parts. However, the static measurement does not take into account the tracking of the voltage integrator's gain by the activity capacitance whereas the dynamic measurements do. This leads to the conclusion that the voltage integrator design successfully reduced the effects of process variations.

Comparing the above variation with the ±3.2% standard deviation for the variation in the M8/M7 cascode current mirror in the voltage integrator (SADMANN, Figure 4.9) and the ±18.1% standard deviation for the variations in the Hamilton synapse [118] (ECDM20), demonstrates the successful performance of the circuits in SADMANN.

7.3. The Performance of the Kohonen Network Running on Analogue Hardware

Of the 4, 5, 9 and the Hopfield/Tank 10 city problems, using weight sets evolved in software and downloaded to FENICS, EPSILON was only able to determine the correct tour order, in the 4, 5 and 9 city cases. The failure to correctly resolve the 10 city tour was due to two reasons:

1 The mis-matches in synaptic multiplications masked small differences between neurons. The software/hardware comparison in Section 6.4 gave the worst case match between the response of one of EPSILON's neurons and its corresponding software neuron as 16.2%. However a mis-match of 5% is more representative of the other neurons' responses (Appendix 10).

2 The output of the Kohonen network only used a third of the available 0-20μs output dynamic range. Thus the full accuracy of EPSILON was not exploited.

A close comparison of the actual and the expected outputs from EPSILON revealed that it was required to differentiate accurately between neuron responses differing by less than 1μs. The Kohonen network was therefore a very tough test of the capabilities of EPSILON, and it was performing very well in determining the 4, 5 and 9 city tours correctly. On the basis of these results EPSILON does not have the very high accuracy required to implement a medium to large scale Kohonen network.

The cycle time to load weights and states to FENICS, and then to read the output states, was about 5s. The main bottleneck was the RS232 link between the host computer and FENICS. Even with the use of a fast data rate (19200 baud), the transfer of 3600 bytes of weight data and 2000 bytes of state data every cycle via the RS232 link was still relatively slow. Thus the 20μs calculation period of EPSILON was insignificant

compared with the system overheads. In the subsequent MLP work by Alister Hamilton [118], the cycle time was improved by connecting the FENICS state bus to the parallel IO card in the IBM PS2. This gave a data transfer rate of 40Kbytes/s for state data and a rate of 20Kbytes/s for weight data, emphasising the point that the performance of any integrated circuit is strongly affected by the architecture of the system around it.

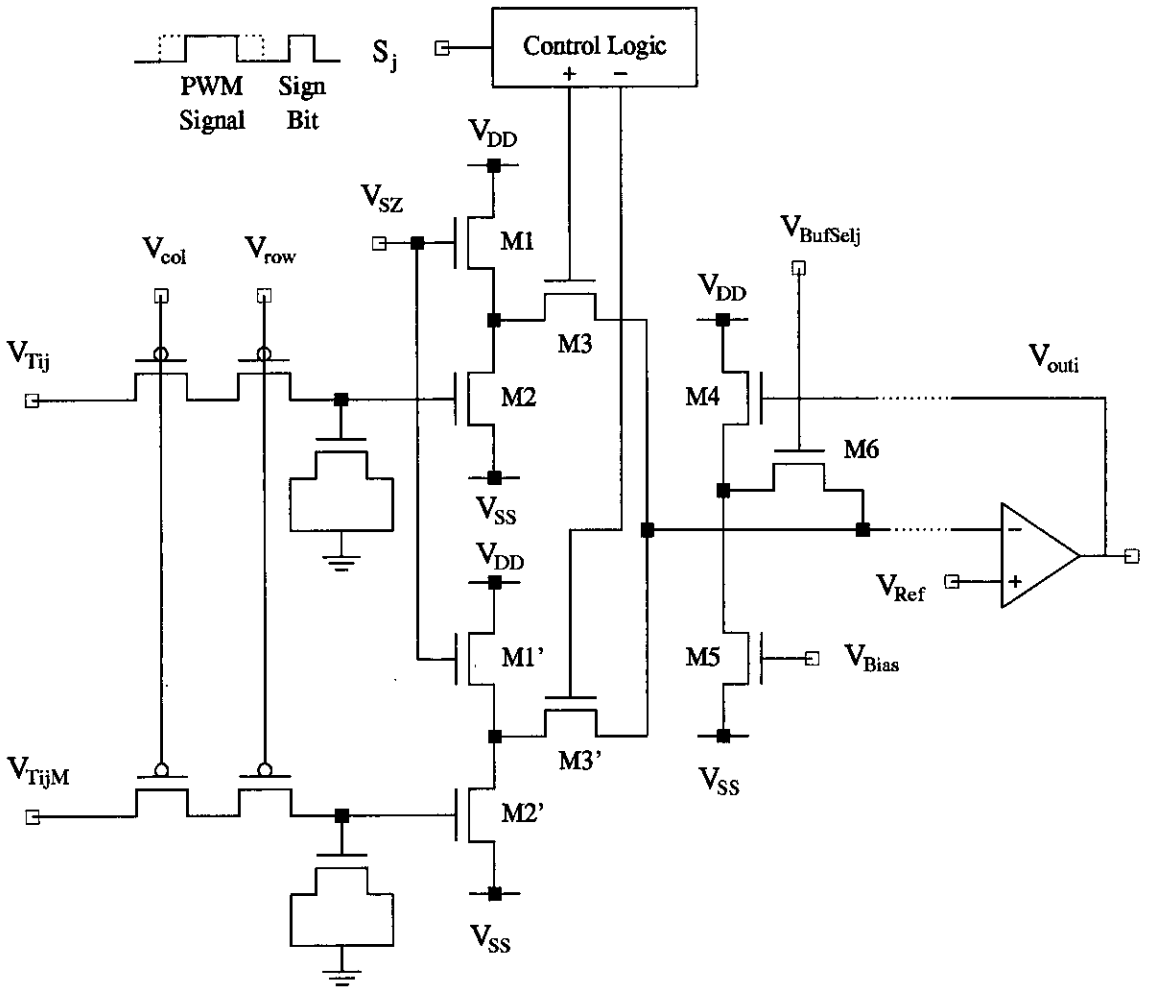


Figure 7.1 Modifications to Distributed Feedback Synapse.

7.4. Future Work

The investigation into the performance of EPSILON running the Kohonen and Hopfield/Tank neural networks is presently incomplete. Further work is required to determine how the Kohonen neural network performs when EPSILON is used during the self-organising phase of the network. To complete the comparison, the performance of the Hopfield/Tank network implemented on EPSILON also needs to be examined.

There are two areas in which the design of the distributed feedback synapse can be improved:

1 Testability.

2 4 Quadrant Multiplication.

The first problem stems from the difficulty, during the testing of EPSILON, to examine the performance of a single synapse. If the buffer stages within a column of distributed feedback synapses are individually switchable (M4,M5 and M6 in Figure 7.1), then the sensitivity of the operational amplifier output voltage to the synaptic weights can be varied. By enabling only one buffer stage within a column the performance of a single synapse can be measured accurately, allowing every synapse to be tested and characterised. The ability to increase the sensitivity of the operational amplifier output allows networks like the Kohonen neural network, which normally use only a fraction of the available output range, to use all of it, thus increasing accuracy and reducing the effects of noise.

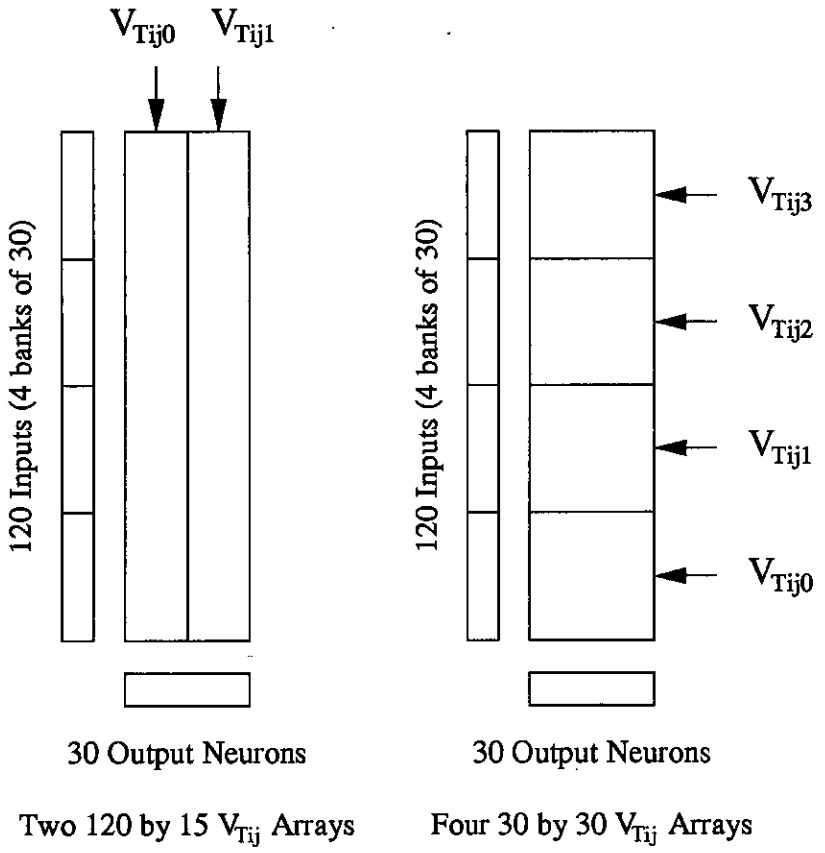


Figure 7.2 A Faster Architecture for Loading Synaptic Weights.

The second limitation of the distributed feedback synapse in its present form is that the synapse only performs 2 quadrant multiplication. This is not a restriction for the Hopfield/Tank, Kohonen or MLP networks as they all have unipolar states, $0 \leq V_j \leq 1.0$. Full 4 quadrant multiplication is however necessary for the calculation of the error signals during the training phase of an MLP neural network. Thus to implement MLP learning algorithms on-chip a full 4 quadrant multiplier circuit is required. A simple 4 quadrant multiplier consists of two transconductance multipliers (Figure 7.1), one driven by the weight voltage, V_{Tij}, and the second by the voltage V_{TijM} which is the "mirror" of V_{Tij} around the zero weight voltage, V_{TijZ} [124]. Due to the linearity of the multipliers, the 2 currents will have the same magnitude but opposite directions. By using the **sign** of the input states to switch in one or other of these 2 currents, 4 quadrant multiplication can be achieved. The sign of the state inputs is encoded as a sign bit which is added as a header to each of the PWM or PFM bit streams stored in the S_j/S_i SRAM. The value of this sign bit is latched within the control logic and directs the following pulse signal to either the positive or negative lines. As only one set of control logic is required per input the extra silicon area is small. The synapse in EPSILON is capable of performing 4 quadrant multiplication by pairing up adjacent input lines and applying the pulse signal to the appropriate line while keeping the other input switched off.

Mis-match between the two transconductance multiplier characteristics due to the effects of process variations will cause the negative and positive phases of the state inputs to become unbalanced. To avoid this problem, the capacitors storing the weight voltage and its mirror voltage can be switched between the same transconductance multiplier. Careful design is required however if charge sharing problems are to be avoided.

The relatively slow cycle time of the FENICS system is due to the large quantities of weight and state data which are transferred between the IBM PS2 and the FENICS board during each cycle [118].

The 3600 element weight array is the largest contributor to this overhead. Downloading this array via the parallel IO card takes 180ms. An additional 3.6ms is required to read the weights into EPSILON. The use of machine code inserts for the data transfer across the parallel IO/state-bus link would double the data transfer rate for weights from 20Kbytes/s to 40Kbytes/s. Even with this increased data rate, 90ms is still required for downloading weights. To improve further on these figures the weight array within EPSILON should be reorganised. At present the 120x30 synaptic array is split into two 120x15 arrays for weighting loading. A more intelligent split would have been to organise EPSILON into four 30x30 arrays, each with an external programming line (Figure 7.2). If the network running on EPSILON requires only 30 inputs, then the 4 banks of 30 inputs and the 4 weight arrays would operate in parallel. The advantage is that now only a 30 by 30 weight array is downloaded, quartering both the data transfer time and the

time to read the weights into EPSILON. Similarly if only 60 of the 120 inputs are needed, the weight load times are halved.

While the use of SRAM to store the pulse signals is simple, it incurs a large data transfer overhead. For example, to load 30 pulse widths down to FENICS, approximately 1Kbyte † of data must be transferred. A much faster way of generating pulse signals would be to use counter based circuits. The 0 to 255 state value loaded into the counter would determine either the pulse frequency or the pulse width of the pulse stream produced by the counter. Similar circuitry can be used on the outputs to convert the output pulse signals into a single byte state value. This would speed up the transfer of state data by a factor of 255. Thus the 25ms presently taken to transfer 1Kbyte of state data via the parallel IO port would be reduced to 0.1ms. The disadvantage is that, as each of the 30 circuits would require three or four 4-bit counters, the number of components required is large. The use of Field Programmable Gate Arrays (FPGAs) would greatly reduce the PCB area.

7.5. Other Work

Appendix 11 outlines work carried out to combine the process invariance of distributed feedback synapse with the fully analogue nature of the transconductance multipliers described in Section 3.3.3. The main advantage of these circuits is that the computation speed is limited only by the speed of the operational amplifiers used. HSPICE simulations showed that a synapse array based on these circuits would settle in 5-10μs. Thus this continuous synapse has a faster computation rate than the distributed feedback synapse while still retaining the theoretical process invariance. Unfortunately, due to funding problems, the testchip containing these circuits was not fabricated.

7.6. Final Remarks

The main objective of the work, to find a means of increasing the synapse's immunity to process variations in comparison with a previous pulse-based synapse [118], was achieved. The distributed feedback synapse performed very well, giving a highly linear multiplication characteristic on a process which was optimised for digital rather than analogue circuits.

The performance of the circuits within EPSILON could be further enhanced by the following alterations:

1 The addition of switchable buffer stages to aid testability and allow the gain of a synaptic column to be varied.

† 30 bit streams x 256 bits of data = 7680 bits ≈ 1KByte

2 The addition of a sign bit to pulse signals and control logic to enable the synapse to perform 4 quadrant multiplication.

3 The use of multiple power buses to reduce the voltage gradient in the synapse power supplies.

4 Better routing of the current set lines for the current mirrors in the feedback amplifiers to reduce the effects of coupling.

Nevertheless, the wide process tolerances did cause inconsistences in the synaptic multiplication. Unless these effects are compensated for, the performance of analogue circuits fabricated on a digital process will be inferior to a software implementation.

When implementing a neural network such as the Kohonen network which requires accurate computation EPSILON's process problems degraded the system's performance. For such a network, a feedback circuit within each synapse would counteract process problems but would greatly increase the synapse area, thus sacrificing the compactness of analogue circuit techniques. It therefore appears that a well setup analogue process is necessary before the circuits in EPSILON can provide the uniformity of multiplication required to perform at software levels.

Neural networks based on learning algorithms such as the delta rule, back propagation and virtual targets are better suited to implementation on EPSILON, as the learning phase of the network will automatically compensate for the effects of process variations. Preliminary results using EPSILON to implement an MLP network for vowel recognition produced more promising results [118]. The virtual targets learning algorithm successfully trained the network to recognise 22 different vowel patterns.

References

1. J.J. Hopfield and D.W. Tank, ""Neural" Computation of Decisions in Optimisation Problems", *Biol. Cybern.*, vol. 52, pp. 141 - 152, 1985.
2. B. Angeniol, G. De La Croix Vaubois, and J. Le Texier, "Self-Organising Feature Maps and the Travelling Salesman Problem", *Neural Networks*, vol. 1, pp. 289 - 293, 1988.
3. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, in *Introduction to Algorithms*, MIT Press, M.A., 1990.
4. G.B. Dantzig, in *Linear Programming and Extensions (Second Edition)*, Princeton University Press, Princeton, NJ, 1963.
5. G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson, "Solution of a Large-Scale Travelling-Salesman Problem", *Operational Research*, vol. 2, pp. 393-410, 1954.
6. R. Fletcher, in *Practical Methods of Optimization (Second Edition)*, John Wiley & Sons, 1987.
7. S. Lin, "Computer Solutions of the Travelling Salesman Problem", *Bell System Technical Journal*, vol. 44, pp. 2245-2269, December, 1965.
8. B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", *Bell System Technical Journal*, vol. 49, pp. 291-307, February, 1970.
9. S. Lin and B.W. Kernighan, "An Effective Heuristic Algorithm for the Travelling Salesman Problem", *Operational Research*, vol. 21, pp. 498-516, 1973.
10. R.M. Brady, "Optimization Strategies Gleaned from Biological Evolution", *Nature*, vol. 317, pp. 804-806, October, 1985.
11. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220, no. 4598, pp. 671-680, May, 1983.
12. S. Kirkpatrick, "Optimization by Simulated Annealing: Quantitative Studies", *Journal of Statistical Physics*, vol. 34, no. 5/6, pp. 975-986, 1984.
13. J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554 - 2558, April, 1982.
14. J.J. Hopfield, "Neural Networks and Physical Systems with Graded Response have Collective Properties like those of Two-State Neurons", *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088 - 3092, May, 1984.

15. G. V. Wilson and G. S. Pawley, "On the Stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank", *Biol. Cybern.*, vol. 58, pp. 63 - 70, 1988.

16. R. A. Paielli, "Simulation Tests of the Optimization Method of Hopfield and Tank Using Neural Networks", NASA Technical Memorandum 101047, November, 1988.

17. Behzad Kamgar-Parsi and Behrooz Kamgar-Parsi, "On Problem Solving with Hopfield Neural Networks", *Submitted to Biological Cybernetics 7/89.*

18. D. W. Tank and J. J. Hopfield, "Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", *IEEE Transactions on Circuits and Systems*, vol. CAS-33, no. 5, pp. 533 - 541, May, 1986.

19. D. W. Tank and J. J. Hopfield, "Collective Computation in Neuronlike Circuits", *Scientific American*, vol. 257, no. 6, pp. 62 - 70, December, 1987.

20. B. C. Levy and M. B. Adams, "Global Optimization with Stochastic Neural Networks", *Proceedings of the IEEE First International Conference on Neural Networks*, vol. III, pp. 681 - 689, San Diego, California, June 21 - 24, 1987.

21. J. H. Cervantes and R. R. Hildebrant, "Comparison of Three Neuron-Based Computation Schemes", *Proceedings of the IEEE First International Conference on Neural Networks*, vol. III, pp. 657 - 671, San Diego, California, June 21 - 24, 1987.

22. D.E. Rumelhart and J.D. McLelland, in *Parallel Distributed Processing : Explorations in the Microstructures of Cognition Volume 1*, MIT Press, 1986.

23. G.E. Hinton, T.J. Sejnowski, and D.H. Ackley, "Bolzmann Machines: Constraint Satisfaction Networks that Learn", *Cognitive Science*, vol. 9, pp. 147 - 169, May, 1984.

24. Y. P. S. Foo and Y. Takefuji, "Stochastic Neural Networks for Solving Job-Shop Scheduling: Part 1. Problem Representation", *Proceedings of the IEEE International Conference on Neural Networks*, vol. II, pp. 275 - 282, San Diego, California, July 24 - 27, 1988.

25. Y. P. S. Foo and Y. Takefuji, "Stochastic Neural Networks for Solving Job-Shop Scheduling: Part 2. Architecture and Simulations", *Proceedings of the IEEE International Conference on Neural Networks*, vol. II, pp. 283 - 290, San Diego, California, July 24 - 27, 1988.

26. K. M. Gutzmann, "Combinational Optimization Using a Continuous State Boltzmann Machine", *Proceedings of the IEEE First International Conference on Neural Networks*, vol. III, pp. 721 - 734, San Diego, California, June 21 - 24, 1987.

27. A. Hemani and A. Postula, "Cell Placement by Self-Organisation", *Neural Networks*, vol. 3, pp. 377 - 383, 1990.

28. A. Hemani and A. Postula, "NISH!E: A neural net inspired scheduling algorithm", *Proceedings of Fourth International Workshop on High Level Synthesis*, Kennebunk, ME, October 15 - 18, 1989.

29. T. Yoshihara, T. Wada, and M.A. Holler, "VLSI Implementations of Learning and Memory Systems: A Review", *Proc. of International Joint Conference on Neural Networks, Seattle*, vol. 1, pp. 993-1000, Morgan Kaufmann, 1991.

30. H. Ritter and T. Kohonen, "Self-Organising Semantic Maps", *Biological Cybernetics*, vol. 61, pp. 241 - 254, 1989.

31. R. P. Lippmann, "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, pp. 4 - 22, April, 1987.

32. T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps", *Biological Cybernetics*, vol. 43, pp. 59 - 69, 1982.

33. R. Durbin and D. Willshaw, "An Analogue Approach to the Travelling Salesman Problem using an Elastic Net Method", *Nature*, vol. 326, no. 6114, pp. 689 - 691, April, 1987.

34. Y. Hirai, "Hardware Implementations of Neural Networks in Japan", *Proc. of the 2nd International Conference on Microelectronics for Neural Networks, Munich*, pp. 435-446, October, 1991.

35. H.P. Graf, E. Sackinger, B. Boser, and L.D. Jackel, "Recent Developments of Electronic Neural Networks in the US and Canada", *Proc. of the 2nd International Conference on Microelectronics for Neural Networks, Munich*, pp. 471-488, October, 1991.

36. N. Morgan, "Digital Implementation of Artifical Neural Networks", *Proc. ITG/IEEE Workshop on Microelectronics for Neural Networks, Dortmund*, pp. 187-195, June, 1990.

37. D. Hammerstrom, "Electronic Implementations of Neural Networks", *Tutorial Notes from the International Joint Conference on Neural Networks, San Diego*, 1992.

38. Motorola Inc, "Motorola's 16, 24 and 32-bit Digital Signal Processing Families", *Product Brochure*, 1991.

39. R.W. Means and L. Lisenbee, "Extensible Linear Floating Point SIMD Neurocomputer Array Processor", *Proc. of International Joint Conference on Neural Networks, Seattle*, vol. 1, pp. 587-592, July, 1991.

40. N. Morgan and et al, "The Ring Array Processor: A Multiprocessing Peripheral for Connectionist Applications", *Journal of Parallel and Distributed Computing*, vol. 14, pp. 248-259, 1992.

41. P. Kohn, J. Bilmes, N. Morgan, and J. Beck, "Software for ANN training on a Ring Array Processor", in *Advances in Neural Information Processing Systems 4*, ed. J.E. Moody, S.J. Hanson and R.P. Lippmann, pp. 781-788, Morgan Kaufmann, 1992.

42. A.F. Murray, A.V.W. Smith, and Z.F. Butler, "Bit - Serial Neural Networks", *Neural Information Processing Systems (NIPS) Conference*, pp. 573-583, 1987.

43. A.F. Murray, Z.F. Butler, and A.V.W. Smith, "VLSI Bit-Serial Neural Networks", in *VLSI for Artificial Intelligence*, ed. J.G.Delgado-Frias, W.R. Moore, pp. 201-208, Kluwer, 1988.

44. A.F. Murray, Z.F. Butler, and A.V.W. Smith, "VLSI Neural Networks", *IEE Colloquium on Parallel Processing*, vol. Dig27, pp. 7/1-4, February, 1988.

45. Z.F. Butler, "A VLSI Hardware Neural Accelerator using Reduced Precision Arithmetic", *PhD. Thesis (University of Edinburgh, UK)*, 1990.

46. F. Aglan, B. Bru, M. Duranton, N. Mauduit, and H. Frydlender, "L-Neuro Boards: Implementing Some Applications", *Proc. of the 2nd International Conference on Microelectronics for Neural Networks, Munich*, pp. 447-453, October, 1991.

47. N. Mauduit, M. Duranton, J. Gobert, and J-A. Sirat, "Lneuro 1.0: A Piece of Hardware LEGO for Building Neural Network Systems", *IEEE Trans. on Neural Networks*, vol. 3, no. 3, pp. 414-422, May, 1992.

48. I. Aleksander and H. Morton, in *Introduction to Neurocomputing*, MIT Press (USA) and North Oxford (UK), 1989.

49. I. Aleksander, "Exploding the Engineering Bottleneck in Neural Computing", *2nd European Seminar on Neural Computing: the Commercial Prospects, London*, February, 1989.

50. A. Masaki, Y. Hirai, and M. Yamada, "Neural Networks in CMOS: a Case Study", *IEEE Circuits and Devices*, vol. 6, no. 4, pp. 13-17, July, 1990.

51. D. Hammerstrom, "A Highly Parallel Digital Architecture for Neural Network Emulation", *Int. Workshop on VLSI for AI and Neural Networks (Oxford)*, 1990.

52. M.S. Melton, T. Phan, D. Reeves, and D. Van den Bout, "VLSI Implementation of TInMANN", in *Advances in Neural Information Processing Systems 3*, ed. R.P. Lippmann, J.E. Moody and D.S. Touretzky, pp. 1046-1052, Morgan Kaufmann, 1991.

53. M.S. Melton, T. Phan, D.S. Reeves, and D.E. Van den Bout, "The TInMANN VLSI Chip", *IEEE Trans. on Neural Networks*, vol. 3, no. 3, pp. 375-384, May, 1992.

54. S. Jones and K. Sammut, "Toroidal Neural Network: Architecture and Processor Granularity Issues", *Proc. ITG/IEEE Workshop on Microelectronics for Neural Networks, Dortmund*, pp. 142-152, June, 1990.

55. S. Jones, K. Sammut, and J. Hunter, "Toroidal Neural Network: Architecture, Operation, Performance", *Proc. of the 2nd International Conference on Microelectronics for Neural Networks, Munich*, pp. 163-170, October, 1991.

56. S. Mackie, "A Flexible, Extensible, Pipelined Architecture for Neural Networks", *4th European Seminar on Neural Computing - Putting Neural Nets to Work, London*, February, 1991.

57. Maxys Circuit Technology Ltd, "A Bit-Serial, Pipelned VLSI Architecture for Pattern Recognition and Neural Networks", *Circuit Architecture Outline*, 1991.

58. J. Beichter, N. Bruls, E. Meister, U. Ramacher, and H. Klar, "Design of A General-Purpose Neural Signal Processor", *Proc. of the 2nd International Conference on Microelectronics for Neural Networks, Munich*, pp. 311-315, October, 1991.

59. U. Ramacher and J. Beichter, "Architecture of a General-Purpose Neural Signal Processor", *Proc. of International Joint Conference on Neural Networks (IJCNN), Seattle*, vol. 1, pp. 443-446, 1991.

60. M. Holler, S. Tam, H. Castro, and R. Benson, "An Electrically Trainable Artificial Neural Network (ETANN) with 10240 "Floating Gate" Synapses", *International Joint Conference on Neural Networks - IJCNN89*, pp. 191-196, June, 1989.

61. Intel Corporation, "80170NX Electrically Trainable Analog Neural Network", *Datasheet*, June, 1991.

62. F.J. Kub, K.K. Moon, I.A. Mack, and F.M. Long, "Programmable Analog Vector-Matrix Multipliers", *IEEE Journal of Solid-State Circuits*, vol. 25, no. 1, pp. 207-214, February, 1990.

63. C. Schneider and H. Card, "CMOS Implementation of Analog Hebbian Synaptic Learning Circuits", *Proc. of International Joint Conference on Neural Networks (IJCNN), Seattle*, vol. 1, pp. 437-442, 1991.

64. S. Satyanarayana, Y. Tsividis, and H.P. Graf, "Analogue Neural Networks with Distributed Neurons", *Electronic Letters*, vol. 25, no. 5, pp. 302-304, March, 1989.

65. S. Satyanarayana, Y. Tsividis, and H.P. Graf, "A Reconfigurable Analogue Neural Network Chip", in *Advances in Neural Information Processing Systems 2*, ed. D.S. Touretzky, pp. 758-768, Morgan Kaufmann, 1990.

66. C. Mead, in *Analog VLSI and Neural Systems*, Addison-Wesley, 1988.

67. C. Mead, "Adaptive Retina", in *Analog VLSI Implementation of Neural Systems*, ed. C. Mead and M. Ismail, Kluwer, 1989.

68. J. Lazzaro and C. Mead, "Circuit Models of Sensory Transduction in the Cochlea", in *Analog VLSI Implementation of Neural Systems*, ed. C. Mead and M. Ismail, Kluwer, 1989.

69. D. Kirk, K. Fleischer, L. Watts, and A. Barr, "Constrained Optimization Applied to the Parameter Setting Problem for Analog Circuits", in *Advances in Neural Information Processing Systems 4*, ed. J.E. Moody, S.J. Hanson and R.P. Lippmann, pp. 789-796, Morgan Kaufmann, 1992.

70. P.B. Denyer and J. Mavor, "Monolithic 256-point Programmable Transversal Filter", *Electronic Letters*, vol. 15, no. 22, pp. 710-712, October, 1979.

71. P.B. Denyer, "Design of Monolithic Programmable Transversal Filters using Charge-Coupled Device Technology", *PhD. Thesis (University of Edinburgh, UK)*, 1980.

72. P.B. Denyer and J. Mavor, "MOST Transconductance Multipliers for Array Applications", *IEE Proc. Pt. 1*, vol. 128, no. 3, pp. 81-86, June, 1981.

73. Il S. Han and Song B. Park, "Voltage-Controlled Linear Resistors by MOS Transistors and their Application to Active RC Filter MOS Integration", *Proc. IEEE*, pp. 1655-1657, November, 1984.

74. P.J. Ryan, D.G. Haigh, M. Banu, and Y. Tsividis, "Fully Integrated Active RC Filters in MOS Technology", *IEEE Journal of Solid-State Circuits*, vol. SC-18, no. 6, pp. 644-651, December, 1983.

75. M. Ismail, S.V. Smith, and R.G. Beale, "A New MOSFET-C Universal Filter Structure for VLSI", *IEEE Journal of Solid-State Circuits*, vol. SC-23, no. 1, pp. 183-194, February, 1988.

76. Y. Tsividis and S. Satyanarayana, "Analogue Circuits for Variable-Synapse Electronic Neural Networks", *Electronic Letters*, vol. 23, no. 24, pp. 302-304, November, 1987.

77. S. Bibyk and M. Ismail, "Issues in Analog VLSI and MOS Techniques for Neural Computing", in *Analog VLSI Implementation of Neural Systems*, ed. C. Mead and M. Ismail, Kluwer, 1989.

78. M. Verleysen, D. Macq, J-P Cornil, and P. Jespers, "Analog Neural Networks with Accurate Sum-of-Products", *ESPRIT - Basic Research Action No 3049, NERVES Report*, 1990.

79. J-P Cornil and P. Jespers, "Analog Vector Multiplier for Integrated Signal Processor Application", *to be published*, 1990.

80. A. Hamilton, A.F. Murray, D.J. Baxter, S. Churcher, H.M. Reekie, and L. Tarassenko, "Integrated Pulse-Stream Neural Networks - Results, Issues and Pointers", *IEEE Trans. Neural Networks*, pp. 385-393, 1992.

81. A. Moopenn, T. Duong, and A.P. Thakoor, "Digital-Analog Hybrid Synapse Chips for Electronic Neural Networks", in *Advances in Neural Information Processing Systems 2*, ed. D.S. Touretzky, pp. 769-776, Morgan Kaufmann, 1990.

82. P.W. Hollis and J.J. Paulos, "Artifical Neural Networks Using MOS Analog Multipliers", *IEEE Journal of Solid-State Circuits*, vol. 25, no. 3, pp. 849-855, June, 1990.

83. M.A. Sivilotti, M.R. Emerling, and C. Mead, "VLSI Architectures for Implementation of Neural Networks", *Proc. AIP Conference on Neural Networks for Computing, Snowbird*, pp. 408-413, 1986.

84. B.E. Boser, E. Sackinger, J. Bromley, Y. Le Cun, and L.D. Jackel, "An Analog Neural Network Processor with Programmable Topology", *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 2017-2025, December, 1991.

85. E. Sackinger, B.E. Boser, J. Bromley, Y. Le Cun, and L.D. Jackel, "Application of the ANNA Neural Network Chip to High-Speed Character Recognition", *IEEE Trans. on Neural Networks*, vol. 3, no. 3, pp. 498-505, May, 1992.

86. E. Sackinger, B.E. Boser, and L.D. Jackel, "A Neurocomputer Board Based on the ANNA Neural Network Chip", in *Advances in Neural Information Processing Systems 4*, ed. J.E. Moody, S.J. Hanson and R.P. Lippmann, pp. 773-780, Morgan Kaufmann, 1992.

87. T.H. Borgstrom, M. Ismail, and S.B. Bibyk, "Programmable Current-Mode Neural Network for Implementation in Analogue MOS VLSI", *IEE Proceedings*, vol. 137, Pt. G, no. 2, April, 1990.

88. M. Verleysen, B. Sirletti, and P. Jespers, "A New CMOS Architecture for Neural Networks", in *VLSI for Artifical Intelligence*, ed. J.G.Delgado-Frias, W.R. Moore, pp. 209-217, Kluwer, 1988.

89. M. Verleysen, B. Sirletti, A.M. Vandemeulebroecke, and P. Jespers, "Neural Networks for High-Storage Content-Addressable Memory: VLSI Circuit and Learning Algorithm", *IEEE Journal of Solid-State Circuits*, vol. 24, no. 3, pp. 562-569, June, 1989.

90. H.P. Graf, L.D. Jackel, R. Janow, D. Henderson, and R. Lee, "Reconfigurable Neural Net Chip with 32K Connections", in *Advances in Neural Information Processing Systems 3*, ed. R.P. Lippmann, J.E. Moody and D.S. Touretzky, pp. 1032-1038, Morgan Kaufmann, 1991.

91. E. Vittoz, H. Oguey, M.A. Maher, O. Nys, E. Dijkstra, and M. Chevroulet, "Analog Storage of Adjustable Synatpic Weights", *Proc. ITG/IEEE Workshop on Microelectronics for Neural Networks, Dortmund*, pp. 69-79, June, 1990.

92. A.J. Agranat, C.F. Neugebauer, R.D. Nelson, and A. Yariv, "The CCD Neural Processor: A Neural Network with 65536 Programmable Analog Synapses", *IEEE Trans. on Circuits and systems*, vol. 37, no. 8, pp. 1073-1075, August, 1990.

93. C.F. Neugebauer and A. Yariv, "A Parallel Analog CCD/CMOS Signal Processor", in *Advances in Neural Information Processing Systems 4*, ed. J.E. Moody, S.J. Hanson and R.P. Lippmann, pp. 748-755, Morgan Kaufmann, 1992.

94. A.A. Reeder, I.P. Thomas, C. Smith, J. Wittgreffe, D. Godfrey, J. Hajto, A. Owen, A.J. Snell, A.F. Murray, M. Rose, and P.G. LeComber, "Application of Analogue Amorphous Silicon Memory Devices to Resistive Synapses for Neural Networks", *International Conference on Neural Networks (Munich)*, pp. 253-260, 1991.

95. H.P. Graf, L.D. Jackel, R.E. Howard, B. Straughn, J.S. Denker, W. Hubbard, D.M. Tennant, and D. Schwartz, "VLSI Implementation of a Neural Network Memory with Several Hundreds of Neurons", *Proc. AIP Conference on Neural Networks for Computing, Snowbird*, pp. 182 - 187, 1986.

96. A.P. Thakoor, J.L. Lamb, A. Moopenn, and J. Lambe, "Binary Synaptic Connections Based on Memory Switching in a-Si:H", in *AIP Conf Proc 151, Neural Networks for Computing, Snowbird*, pp. 426-431, New York, 1986.

97. Y. Arima, K. Mashiko, K. Okada, T. Yamada, A. Maeda, H. Notani, H. Kondoh, and S. Kayano, "A 336-Neuron, 28K-Synapse, Self Learning Neural Network Chip with Branch-Neuron-Unit Architecture", *IEEE Journal of Solid-State Circuits*, vol. 26, no. 11, pp. 1637-1643, November, 1991.

98. J. Alspector, J.W. Gannett, S. Haber, M.B. Parker, and R. Chu, "A VLSI-Efficient Technique for Generating Multiple Uncorrelated Noise Sources and its Application to Stochastic Neural Networks", *IEEE Trans. Circuits and Systems*, vol. 38, no. 1, pp. 109-123, January, 1991.

99. A.F. Murray and A.V.W. Smith, "Asynchronous Arithmetic for VLSI Neural Systems", *Electronics Letters*, vol. 23, no. 12, pp. 642-643, June, 1987.

100. A.F. Murray and A.V.W. Smith, "Asynchronous VLSI Neural Networks using Pulse Stream Arithmetic", *IEEE Journal of Solid-State Circuits and Systems*, vol. 23, no. 3, pp. 688-697, 1988.

101. M.S. Tomlinson, D.J. Walker, and M.A. Sivilotti, "A Digital Neural Network Architecture for VLSI", *Proc. of International Joint Conference on Neural Networks (IJCNN), San Diego*, vol. 2, pp. 545-550, 1990.

102. H. Eguchi, T. Furuta, H. Horiguchi, S. Oteki, and T. Kitaguchi, "Neural Network LSI with On-Chip Learning", *Proc. of International Joint Conference on Neural Networks (IJCNN), Seattle*, vol. 1, pp. 453-456, 1991.

103. A. Siggelkow, A.J. Beltman, J.A.G. Nijhuis, and L. Spaanenburg, "Pulse-Density Modulated Neural Networks on a Semi-Custom Gate Forest", *Proc. ITG/IEEE Workshop on Microelectronics for Neural Networks, Dortmund (Germany).*, pp. 16-27, June 1990.

104. Y. Tsividis and D. Anastassiou, "Switched-Capacitor Neural Networks", *Electronic Letters*, vol. 23, no. 18, pp. 958-959, August, 1987.

105. M.J. Brownlow, "An Analogue VLSI Neural Network", *MSc Project Report MSP 77, Unversity of Edinburgh*, August, 1989.

106. M. Brownlow, L. Tarassenko, and A.F. Murray, "Analogue Computation using VLSI Neural Network Devices", *Electronics Letters*, vol. 26, no. 16, pp. 1297-1299, August, 1990.

107. M. Brownlow, L. Tarassenko, and A.F. Murray, "Results from Pulse Stream Neural Network Devices", *Proc. Int. Workshop on VLSI for Artificial Intelligence and Neural Networks*, pp. A2/1-A2/11, September, 1990.

108. J.E. Tomberg and K.K.K. Kaski, "Pulse-Density Modulation Technique in VLSI Implementations of Neural Network Algorithms", *IEEE Journal of Solid-State Circuits*, vol. 25, no. 5, October, 1990.

109. J. Tomberg, "Integrated Circuit Implementations of Artifical Neural Networks", *PhD. Thesis (Tampere Uinversity of Technology, Finland)*, 1992.

110. L.M. Reyneri and M. Sartori, "A Neural Vector Matrix Multiplier using Pulse Width Modulation Techniques", *Proc. of the 2nd International Conference on Microelectronics for Neural Networks, Munich*, pp. 269-272, October, 1991.

111. D. Del Corso, F. Gregoretti, C. Pellegrini, and L.M. Reyneri, "An Artificial Neural Network Based on Multiplexed Pulse Streams", *Proc. ITG/IEEE Workshop on Microelectronics for Neural Networks, Dortmund*, pp. 28-39, June, 1990.

112. A. Hamilton, A.F. Murray, S. Churcher, and H.M. Reekie, "Working Analogue Pulse Stream Neural Network Chips", *Proc. Int. Workshop on VLSI for Artificial Intelligence and Neural Networks*, pp. A3/1-A3/9, September, 1990.

113. A.F. Murray, L. Tarassenko, and A.V.W. Smith, "Fully-Programmable Analogue VLSI Devices for the Implementation of Neural Networks", in *VLSI for Artificial Intelligence*, ed. J.G.Delgado-Frias, W.R. Moore, pp. 236-244, Kluwer, 1988.

114. A.F. Murray, L. Tarassenko, and A. Hamilton, "Programmable Analogue Pulse-Firing Neural Networks", *Neural Information Processing Systems (NIPS) Conference*, pp. 671-677, Morgan Kaufmann, 1989.

115. A.F. Murray, M. Brownlow, A. Hamilton, Il Song Han, H.M. Reekie, and L. Tarassenko, "Pulse-Firing Neural Chips for Hundreds of Neurons", *Neural Information Processing Systems (NIPS) Conference*, pp. 785-792, Morgan Kaufmann, 1990.

116. P.E. Allen and D.R. Holberg, in *CMOS Analog Circuit Design*, Holt, Rinehart and Winston, New York, 1987.

117. S. Churcher, "VLSI Neural Networks for Computer Vision", *PhD. Thesis (University of Edinburgh, UK)*, 1993.

118. A. Hamilton, "Analogue VLSI Circuits for Neural Net Implementations", *PhD. Thesis (University of Edinburgh, UK)*, 1993.

119. E. Vittoz, "Analog VLSI Implementations of Neural Networks", in *Journees d'Electroniques 1989*, Presses Polytechniques Romandes, Lausanne, 1989.

120. *Solo 1200 Analogue and Block Databook (ES2-014-0057)*, European Silicon Structures, March, 1988.

121. *Solo 1400 Databook - ECPD15 Library (ES2-014-0201)*, European Silicon Structures, January, 1991.

122. J. Tombs, Personal Communication, October, 1992.

123. A.F. Murray and P.J. Edwards, "Synaptic Weight Noise During MLP Training Enhances Fault Tolerance", *Neural Information Processing Systems (NIPS) Conference*, 1992.

124. R. Woodburn, Personal Communication, October, 1992.

Appendix 1

Results for the 100 Randomly Generated 10 City Data Sets

City Set	Brute Force			Hopfield/Tank			Kohonen
	Min	Mean	Max	Min	Mean	Max	
city10_0	2.691	4.760	6.288	2.691	2.934	3.949	2.691
ran10_00	2.434	3.886	4.920	2.434	2.581	3.151	2.450
ran10_01	2.743	4.366	5.433	2.743	2.955	3.323	2.743
ran10_02	3.300	5.758	7.718	3.310	3.523	4.126	3.346
ran10_03	2.809	5.088	6.722	2.809	3.154	4.313	2.889
ran10_04	2.516	4.978	6.688	2.516	2.856	4.046	2.516
ran10_05	2.986	5.490	7.209	2.986	3.210	4.073	2.986
ran10_06	3.290	5.898	7.706	3.394	3.583	3.939	3.290
ran10_07	3.409	5.963	7.764	3.409	3.594	4.495	3.409
ran10_08	3.246	6.081	8.185	3.252	3.437	4.170	3.252
ran10_09	3.203	5.374	6.936	3.262	3.455	3.892	3.203

City Set	Brute Force			Hopfield/Tank			Kohonen
	Min	Mean	Max	Min	Mean	Max	
ran10_10	2.897	5.153	6.807	2.897	3.219	3.896	2.897
ran10_11	2.888	5.091	6.714	2.888	3.226	3.957	2.888
ran10_12	2.929	5.490	7.273	3.055	3.276	3.954	2.929
ran10_13	2.624	4.239	5.375	2.624	2.812	3.101	2.624
ran10_14	3.161	5.130	6.543	3.237	3.420	3.576	3.161
ran10_15	2.635	4.065	5.135	2.635	2.923	3.354	2.635
ran10_16	3.047	5.252	6.680	3.047	3.363	3.737	3.047
ran10_17	2.469	4.939	6.772	2.503	2.705	3.169	2.477
ran10_18	3.400	6.082	8.156	3.431	3.670	4.065	3.400
ran10_19	2.955	4.853	6.140	2.955	3.261	3.813	3.176
ran10_20	2.634	5.116	6.893	2.689	2.961	3.289	2.648
ran10_21	3.426	5.714	7.364	3.590	3.693	3.749	3.543
ran10_22	1.758	3.367	4.479	1.758	2.113	2.739	1.758
ran10_23	2.433	4.523	6.028	2.433	2.786	3.302	2.433
ran10_24	2.406	4.540	5.914	2.406	2.658	4.205	2.406
ran10_25	3.357	5.914	7.806	3.357	3.460	3.666	3.357
ran10_26	2.753	5.090	6.738	2.753	3.054	3.378	2.866
ran10_27	2.732	5.433	7.309	2.732	3.030	3.280	2.782
ran10_28	2.885	6.120	8.483	2.885	3.273	4.012	2.885
ran10_29	3.547	5.697	7.239	3.580	3.680	3.780	3.579
ran10_30	2.810	4.709	6.177	3.149	3.155	3.160	2.818
ran10_31	2.657	5.406	7.318	2.657	2.844	3.292	2.657
ran10_32	3.501	5.747	7.253	-	-	-	3.501
ran10_33	3.004	5.408	7.176	3.004	3.203	3.895	3.004
ran10_34	2.579	5.066	6.639	2.768	2.885	3.005	2.579
ran10_35	2.911	5.122	6.817	2.911	2.999	3.295	2.911
ran10_36	2.575	4.265	5.290	2.575	2.869	3.452	2.590
ran10_37	2.397	4.565	6.033	2.397	2.703	3.239	2.397
ran10_38	2.881	5.106	6.896	3.051	3.279	3.748	3.051
ran10_39	2.263	4.163	5.601	2.263	2.563	2.994	2.263

City Set	Brute Force			Hopfield/Tank			Kohonen
	Min	Mean	Max	Min	Mean	Max	
ran10_40	2.853	4.891	6.250	2.854	3.006	3.270	2.872
ran10_41	3.518	5.211	6.471	3.631	3.631	3.631	3.518
ran10_42	3.302	5.809	7.679	3.385	3.654	3.970	3.302
ran10_43	3.193	5.914	7.945	3.193	3.324	3.914	3.193
ran10_44	2.496	4.538	5.990	2.496	2.670	2.845	2.496
ran10_45	3.527	6.840	9.196	3.527	3.585	3.948	3.527
ran10_46	2.339	4.506	6.048	2.339	2.616	3.414	2.339
ran10_47	3.323	5.229	6.443	3.340	3.482	3.634	3.340
ran10_48	2.905	5.725	7.714	2.905	3.082	3.392	2.905
ran10_49	3.274	5.418	7.063	3.274	3.589	4.272	3.335
ran10_50	2.495	5.051	6.875	2.495	2.774	3.210	2.495
ran10_51	2.463	4.573	6.163	2.463	2.675	3.144	2.463
ran10_52	1.828	3.312	4.331	1.828	2.171	2.751	1.828
ran10_53	2.477	3.836	4.872	2.751	2.757	2.763	2.477
ran10_54	3.004	4.878	6.256	3.012	3.300	3.673	3.004
ran10_55	3.030	5.109	6.607	3.075	3.439	3.840	3.055
ran10_56	2.722	5.178	7.031	2.722	3.018	3.517	2.722
ran10_57	3.298	5.261	6.812	3.354	3.412	3.445	3.354
ran10_58	3.001	5.613	7.417	3.001	3.314	4.419	3.001
ran10_59	2.607	4.405	5.595	2.647	2.853	3.113	2.628
ran10_60	3.286	5.318	6.700	3.342	3.600	3.960	3.343
ran10_61	2.817	4.655	5.911	2.817	3.151	3.366	2.817
ran10_62	3.346	5.653	7.273	3.346	3.698	4.449	3.346
ran10_63	2.647	4.154	5.286	2.647	2.857	3.457	2.647
ran10_64	2.411	4.697	6.259	2.411	2.654	3.203	2.411
ran10_65	2.560	4.248	5.374	2.631	2.937	3.245	2.560
ran10_66	2.861	4.821	6.377	2.873	3.207	3.541	2.861
ran10_67	2.704	4.563	6.050	2.704	2.931	3.579	2.852
ran10_68	2.485	4.718	6.185	2.485	2.743	3.111	2.485
ran10_69	2.749	5.691	7.886	2.775	3.261	4.054	2.749

City Set	Brute Force			Hopfield/Tank			Kohonen
	Min	Mean	Max	Min	Mean	Max	
ran10_70	3.206	5.610	7.355	3.206	3.421	3.973	3.206
ran10_71	2.589	4.029	5.115	2.613	2.862	3.420	2.589
ran10_72	2.907	5.883	8.033	-	-	-	2.926
ran10_73	2.515	4.804	6.491	2.515	2.704	3.588	2.635
ran10_74	2.512	4.654	6.260	2.512	2.873	3.561	2.512
ran10_75	3.105	6.168	8.349	3.105	3.358	4.347	3.105
ran10_76	2.882	5.026	6.515	2.901	3.123	3.443	2.882
ran10_77	3.415	5.165	6.661	-	-	-	3.415
ran10_78	3.062	5.340	6.881	3.084	3.365	3.610	3.062
ran10_79	3.252	5.887	7.670	3.252	3.476	3.987	3.252
ran10_80	3.325	5.860	7.556	3.326	3.477	3.998	3.326
ran10_81	2.684	5.014	6.602	2.684	2.930	3.412	2.736
ran10_82	2.621	5.112	6.860	2.621	2.829	3.281	2.621
ran10_83	2.857	4.946	6.508	2.857	3.107	3.825	3.021
ran10_84	2.893	5.986	8.081	2.893	3.105	3.662	2.893
ran10_85	2.981	5.140	6.735	2.981	3.228	4.238	2.981
ran10_86	3.016	5.382	7.088	3.050	3.188	3.813	3.016
ran10_87	2.937	5.307	7.127	2.937	3.093	3.763	2.937
ran10_88	3.104	5.501	7.338	3.104	3.402	3.926	3.104
ran10_89	2.768	5.636	7.639	2.768	3.158	3.914	2.768
ran10_90	2.707	5.286	7.184	2.707	3.126	3.748	2.707
ran10_91	3.170	6.072	8.138	3.170	3.521	4.272	3.170
ran10_92	2.694	4.680	6.165	2.694	2.990	3.640	2.694
ran10_93	2.215	4.366	5.899	2.215	2.535	3.155	2.215
ran10_94	3.226	5.517	7.167	3.226	3.368	3.687	3.226
ran10_95	3.072	5.033	6.381	3.072	3.482	4.248	3.072
ran10_96	2.715	4.966	6.436	2.715	3.011	3.621	2.715
ran10_97	2.048	3.485	4.538	2.048	2.443	2.868	2.048
ran10_98	2.428	4.744	6.418	2.428	2.756	3.305	2.428
ran10_99	2.566	4.420	5.655	2.566	2.800	3.124	2.566

Appendix 2

The Stirling's Approximation for Factorials of Large Numbers

For numbers larger than 69 approximations have to be made to estimate the value of the factorials of large positive numbers. The Gamma function of (n+1) is equal to n factorial.

$$n! = \Gamma(n+1)$$

Using the Stirling's Approximation to the Gamma function [1] gives

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Example : 1000!

$$\begin{aligned} 1000! &= \sqrt{2000\pi}\left(\frac{1000}{e}\right)^{1000} \\ &= 79.267(367.88)^{1000} \\ &\approx 100^{1000} \\ &\approx 10^{2000} \end{aligned}$$

References

1. E. Kreyszig, in *Advanced Engineering Mathematics (Fifth Edition)*, John Wiley & Sons, 1983.

Appendix 3

On-Chip Learning Implementations

The heavy area penalty incurred by distributing signals globally within a synaptic array encourages the implementation of learning algorithms which use local information such as the presynaptic and postsynaptic states to modify a synapse's weight. As a result most of the implementations of on-chip learning use local algorithms such as Hebbian and Boltzmann learning rather than Back-Propagation which requires error signals to be passed between layers. The use of Boltzmann learning further simplifies the implementation as it has binary neural states.

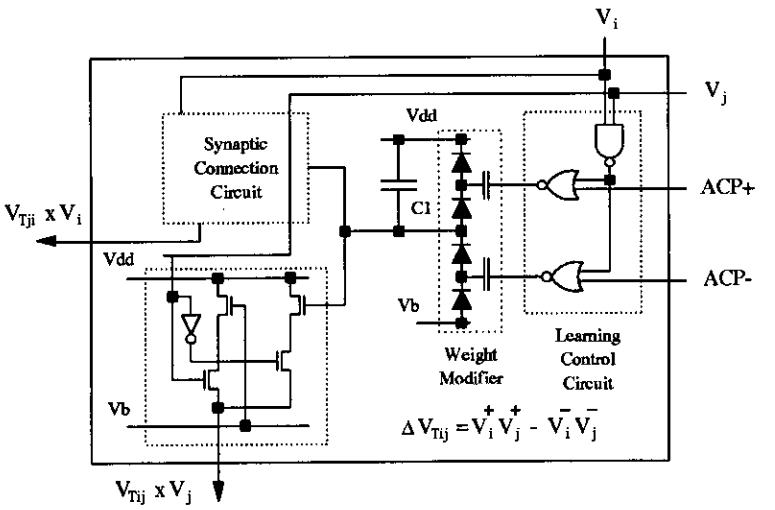


Figure A3.1 Schematic of Arima *et al* Synapse Cell

A3.1 Mitsubishi Boltzmann

Arima *et al* [1, 2] have recently implemented a very large Boltzmann neural network in silicon. The combination of 1μm CMOS technology and the Boltzmann algorithm's binary neural states, enabled them to squeeze 336 neurons and 28000 synapses into a 14.5mm by 14.5mm die size. The Branch-Neuron-Unit (BNU) architecture used enables 200 such devices to be cascaded together creating a true, massively parallel, neural network.

To keep the synapse design simple (Figure A3.1) the mean field approximation of the Boltzmann weight update algorithm is used.

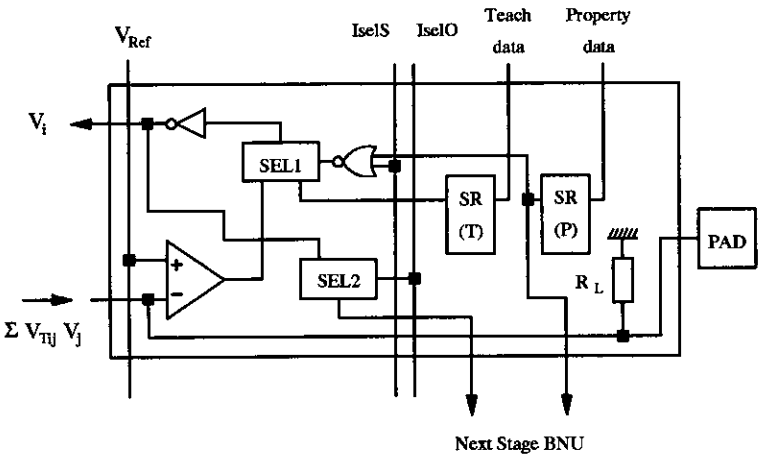


Figure A3.2 Schematic of Arima *et al* Branch Neuron Unit

$$\Delta V_{Tij} = \eta\left(V_i^+ V_j^+ - V_i^- V_j^-\right) \qquad \text{(A3.1)}$$

Two charge pump circuits either increment or decrement the voltage stored on the the weight capacitor, C1, under the control of the above equation. The learning is carried out in 2 distinct phases. During the first phase the input and output neurons are set to their desired values, V_j^+ and V_i^+. The NAND gate in the learning control circuit carries out the required multiplication states, the result of which gates the pulses applied to the ACP + line. During the second phase of learning inputs V_j^-, remain fixed but the outputs are allowed to settle to their own values, V_i^-. Again the NAND gate carries out the required state multiplication but this time it is the ACP – line which is pulsed, decrementing the weight voltage as appropriate. Thus the net weight change is as defined in Equation A3.1. The resultant weight voltage controls the current of 2 current sources, representing the weights V_{Tij} and V_{Tji} respectively. Their currents are switched in and out by V_i and V_j to give the desired $V_{Tij} \times V_j$ and $V_{Tji} \times V_i$ multiplications.

The sum of the synaptic currents within a row or column is converted into a voltage via the resistor R_L in the Branch Neuron Unit (Figure A3.2). The neural activity obtained forms one input to the comparator. The second input, V_{Ref}, determines the neuron's threshold voltage. By applying a damped oscillating signal to this input, simulated annealing can be implemented. Unfortunately, due to the presence of only 4 V_{Ref} lines per device, many of the neurons will have exactly the same noise source and as a result will be correlated. Arima *et al* are presently researching the implications of this pseudosimulated annealing process.

A major limitation of this chip is that storing the learned weight capacitively requires the chip to be retrained on a regular basis to correct the capacitive decay of the weights. Also, the size of the weight increments is about 10% of their full range values

which may not be accurate enough for stable convergence in some multilayer network architectures.

A3.2 Bellcore Boltzmann

Alspector *et al* [3-6] have concentrated on developing multiple, on-chip, uncorrelated noise sources based on a single linear feedback shift register (LFSR). The use of one LFSR per chip rather than one per neuron allows a much higher level of integration on a single chip. A cascadable 32-neuron chip containing 496 bidirectional synapses has been fabricated. The synapses store the 5-bit weights digitally on flip-flops. These flip-flops are driven from an up/down counter which is incremented/decremented according to the correlation between the neuron states that intersect at that synapse. The digital weights are converted into a current via circuitry similar to the MDAC circuitry mentioned in Section 2.3.4. As in the Arima chip there are two phases during learning, teacher and student.

The chip has successfully solved the parity and input replication problems achieving a computation rate of 100MCPS for both learning and evaluation. In general, the limited 5-bit accuracy of the synaptic weights did not affect the performance of the network.

A3.3 Manitoba Hebbian

The wide-range Gilbert multiplier is the basis of the Hebbian and mean field learning chips designed by Schneider *et al* [7-10]. As the analogue weight voltages are stored on a capacitor, weight decay is a problem but the consequent need to retrain periodically is promoted as an advantage since it should compensate for drifts in circuit parameters.

References

1. Y. Arima, K. Mashiko, K. Okada, T. Yamada, A. Maeda, H. Kondoh, and S. Kayano, "A Self Learning Neural Network Chip with 125 Neurons and 10K Self-organization Synapses", *IEEE Journal of Solid-State Circuits*, vol. 26, no. 4, pp. 607-611, April, 1991.
2. Y. Arima, K. Mashiko, K. Okada, T. Yamada, A. Maeda, H. Notani, H. Kondoh, and S. Kayano, "A 336-Neuron, 28K-Synapse, Self Learning Neural Network Chip with Branch-Neuron-Unit Architecture", *IEEE Journal of Solid-State Circuits*, vol. 26, no. 11, pp. 1637-1643, November, 1991.
3. J. Alspector, B. Gupta, and R.B. Allen, "Performance of a Stochastic Learning Microchip", in *Advances in Neural Information Processing Systems 1*, ed. D.S. Touretzky, pp. 748-760, Morgan Kaufmann, 1989.

4. J. Alspector, "Neural-Style Microsystems that Learn", *IEEE Communications Magazine*, pp. 29-36, November, 1989.

5. J. Alspector, R.B. Allen, A. Jayakumar, T. Zeppenfeld, and R. Meir, "Relaxation Networks for Large Supervised Learning Problems", in *Advances in Neural Information Processing Systems 3*, ed. R.P. Lippmann, J.E. Moody and D.S. Touretzky, pp. 1015-1021, Morgan Kaufmann, 1991.

6. J. Alspector, J.W. Gannett, S. Haber, M.B. Parker, and R. Chu, "A VLSI-Efficient Technique for Generating Multiple Uncorrelated Noise Sources and its Application to Stochastic Neural Networks", *IEEE Trans. Circuits and Systems*, vol. 38, no. 1, pp. 109-123, January, 1991.

7. C. Schneider and H. Card, "CMOS Implementation of Analog Hebbian Synaptic Learning Circuits", *Proc. of International Joint Conference on Neural Networks (IJCNN), Seattle*, vol. 1, pp. 437-442, 1991.

8. C. Schneider and H. Card, "Analog VLSI Models of Mean Field Networks", in *VLSI for Artificial Intelligence*, ed. J.G.Delgado-Frias, W.R. Moore, pp. 185-194, Plenum, 1992.

9. C. Schneider and H. Card, "Analogue CMOS Hebbian Synapses", *Electronic Letters*, vol. 27, no. 9, pp. 785-786, April, 1991.

10. C.R. Schneider and H.C. Card, "CMOS Mean Field Learning", *Electronic Letters*, vol. 27, no. 19, pp. 1702-1704, September, 1991.

Appendix 4

Static Measurements of SADMANN 1010PI

V_{Tij} (V)	$V_{outi} - V_{0Z}$ (V)					
	HSPICE L2-Typ	Min	Mean	Max	Variation	Std Dev
0.00	-1.11	-1.13	-1.06	-1.02	+7.5%/-10.8%	±3.6%
0.25	-1.11	-1.12	-1.06	-1.01	+8.8%/-9.4%	±3.5%
0.50	-1.11	-1.12	-1.06	-1.01	+8.3%/-9.9%	±3.5%
0.75	-1.11	-1.12	-1.06	-1.01	+8.1%/-10.1%	±3.5%
1.00	-1.11	-1.11	-1.06	-1.01	+7.8%/-8.8%	±3.5%
1.25	-1.11	-1.06	-1.00	-0.96	+7.3%/-9.3%	±3.3%
1.50	-1.03	-0.97	-0.93	-0.89	+6.5%/-6.8%	±2.8%
1.75	-0.90	-0.85	-0.81	-0.78	+4.3%/-7.3%	±2.5%
2.00	-0.75	-0.72	-0.68	-0.65	+4.5%/-7.1%	±2.3%
2.25	-0.61	-0.59	-0.55	-0.53	+3.1%/-6.8%	±2.2%
2.50	-0.49	-0.47	-0.44	-0.41	+4.3%/-5.6%	±2.1%
2.75	-0.38	-0.36	-0.32	-0.30	+4.0%/-6.0%	±2.4%
3.00	-0.26	-0.25	-0.21	-0.18	+5.5%/-6.1%	±2.4%
3.25	-0.14	-0.15	-0.10	-0.07	+5.1%/-8.1%	±2.5%
3.50	-0.03	-0.04	0.00	0.04	+6.1%/-7.1%	±2.6%
3.75	0.08	0.06	0.11	0.14	+5.5%/-7.8%	±2.7%
4.00	0.20	0.16	0.21	0.25	+7.0%/-7.9%	±2.8%
4.25	0.31	0.26	0.31	0.35	+7.0%/-7.9%	±3.0%
4.50	0.42	0.35	0.41	0.45	+6.8%/-9.8%	±3.1%
4.75	0.53	0.45	0.51	0.55	+7.3%/-9.3%	±3.3%
5.00	0.64	0.54	0.60	0.66	+9.3%/-10.6%	±3.4%

Table A4.1 The Percentage Variations of the V_{Tij} to $V_{outi} - V_{OZ}$ Characteristic. All percentages are taken relative to the mean response for V_{Tij}=5.00V (0.60V).

$V_{outi} - V_{OZ}$ (V)	I_i (μA)					
	HSPICE L2-Typ	Min	Mean	Max	Variation	Std Dev
-1.00	-0.47	-0.49	-0.46	-0.43	+6.3%/-6.6%	±3.2%
-0.90	-0.45	-0.47	-0.43	-0.40	+7.0%/-7.2%	±3.3%
-0.80	-0.41	-0.43	-0.39	-0.36	+6.8%/-8.5%	±3.5%
-0.70	-0.37	-0.39	-0.34	-0.32	+6.1%/-8.7%	±3.2%
-0.60	-0.33	-0.33	-0.29	-0.26	+6.6%/-9.4%	±3.6%
-0.50	-0.28	-0.27	-0.22	-0.19	+7.0%/-9.8%	±3.9%
-0.40	-0.23	-0.20	-0.15	-0.12	+7.2%/-10.0%	±4.1%
-0.30	-0.17	-0.13	-0.08	-0.04	+8.5%/-10.9%	±4.4%
-0.20	-0.12	-0.06	-0.00	0.04	+9.4%/-11.8%	±4.8%
-0.10	-0.06	0.02	0.08	0.12	+10.0%/-12.7%	±5.1%
0.00	0.0	0.10	0.16	0.21	+10.9%/-13.5%	±5.4%
0.10	0.06	0.17	0.24	0.29	+11.8%/-14.2%	±5.8%
0.20	0.12	0.25	0.31	0.37	+12.4%/-14.6%	±6.0%
0.30	0.17	0.32	0.39	0.45	+13.1%/-15.3%	±6.4%
0.40	0.23	0.39	0.46	0.52	+14.2%/-15.7%	±6.9%
0.50	0.28	0.45	0.53	0.59	+14.6%/-16.4%	±7.0%
0.60	0.33	0.51	0.58	0.65	+15.1%/-16.6%	±7.2%
0.70	0.37	0.55	0.63	0.70	+15.3%/-17.0%	±7.4%
0.80	0.41	0.58	0.67	0.74	+15.5%/-17.5%	±7.4%
0.90	0.45	0.61	0.69	0.76	+15.1%/-18.3%	±7.3%
1.00	0.47	0.63	0.71	0.79	+16.8%/-18.1%	±7.8%

Table A4.2 The Percentage Variations of the ($V_{outi} - V_{OZ}$) to I_i Characteristic. All percentages are taken relative to the mean response for ($V_{Tij} - V_{OZ}$)=-1.00V (-0.46μA).

Appendix 5

Dynamic Measurements of SADMANN 1010PI

V_{Tij} (V)	dX_i/dt (V/ms)					Measurement Accuracy	
	Min	Mean	Max	Variation	Std Dev	Variation	Std Dev
1.11	-8.36	-7.67	-6.79	+17.6%/-14.1%	±6.7%	+4.2%/-3.2%	±1.4%
1.31	-8.31	-7.67	-6.90	+15.4%/-13.0%	±6.6%	+4.0%/-3.5%	±1.6%
1.50	-8.23	-7.52	-6.74	+15.8%/-14.3%	±6.4%	+3.9%/-3.6%	±1.6%
1.70	-7.64	-6.98	-6.28	+14.1%/-13.2%	±6.3%	+3.8%/-2.9%	±1.5%
1.89	-6.87	-6.25	-5.58	+13.6%/-12.6%	±6.0%	+3.5%/-3.0%	±1.4%
2.08	-6.12	-5.45	-4.84	+12.2%/-13.7%	±5.6%	+3.0%/-3.7%	±1.4%
2.28	-5.28	-4.66	-4.05	+12.3%/-12.4%	±5.2%	+4.2%/-3.1%	±1.8%
2.47	-4.46	-3.90	-3.31	+11.9%/-11.2%	±4.8%	+3.0%/-3.9%	±1.6%
2.67	-3.71	-3.15	-2.61	+10.8%/-11.2%	±4.6%	+3.2%/-2.9%	±1.3%
2.86	-2.91	-2.43	-1.90	+10.8%/-9.5%	±4.2%	+2.5%/-3.4%	±1.2%
3.06	-2.27	-1.72	-1.24	+9.8%/-11.1%	±4.1%	+2.5%/-3.3%	±1.2%
3.25	-1.52	-1.03	-0.48	+11.1%/-9.9%	±4.1%	+14.7%/-2.9%	±3.1%
3.45	-0.90	-0.37	-0.03	+6.9%/-10.7%	±3.6%	+2.4%/-3.0%	±1.2%
3.64	0.12	0.45	1.13	+13.6%/-6.8%	±3.5%	+2.9%/-2.4%	±1.1%
3.84	0.56	1.08	1.77	+13.9%/-10.5%	±4.4%	+3.3%/-2.6%	±1.2%
4.03	1.16	1.75	2.48	+14.7%/-11.9%	±4.6%	+2.7%/-3.8%	±1.3%
4.22	1.81	2.40	3.23	+16.6%/-12.0%	±4.7%	+3.1%/-3.5%	±1.4%
4.42	2.43	3.08	3.86	+15.7%/-13.1%	±5.0%	+5.9%/-5.3%	±2.8%
4.61	3.26	3.72	4.41	+13.9%/-9.3%	±4.5%	+4.2%/-7.8%	±1.9%
4.81	3.79	4.32	5.05	+14.6%/-10.8%	±4.8%	+2.8%/-6.6%	±1.7%
5.00	4.35	4.95	5.75	+16.0%/-12.3%	±5.2%	+4.2%/-6.0%	±1.7%

Table A5.1 The Percentage Variations of the V_{Tij} to dX_i/dt Characteristic (S_j=50%). All percentages are taken relative to the mean response for V_{Tij}=5.00V and S_j=50%.

S_j (%)	dX_i/dt (V/ms)					Measurement Accuracy	
	Min	Mean	Max	Variation	Std Dev	Variation	Std Dev
0	-1.80	-1.09	-0.61	+9.7%/-14.3%	±4.0%	+2.8%/-2.6%	±1.0%
2	-1.97	-1.28	-0.79	+9.9%/-14.0%	±4.0%	+2.5%/-2.3%	±1.1%
4	-2.18	-1.46	-0.91	+11.1%/-14.4%	±4.0%	+3.0%/-2.2%	±1.1%
6	-2.43	-1.67	-1.15	+10.4%/-15.5%	±4.2%	+2.1%/-2.2%	±1.0%
8	-2.62	-1.85	-1.36	+9.8%/-15.7%	±4.2%	+2.8%/-2.7%	±1.1%
10	-2.82	-2.04	-1.52	+10.5%/-15.7%	±4.2%	+2.7%/-2.5%	±1.1%
12	-3.03	-2.24	-1.74	+10.1%/-16.0%	±4.3%	+2.3%/-2.5%	±1.1%
14	-3.31	-2.45	-1.91	+11.0%/-17.4%	±4.4%	+3.1%/-3.3%	±1.2%
16	-3.47	-2.61	-2.11	+10.2%/-17.3%	±4.5%	+3.1%/-3.4%	±1.3%
18	-3.67	-2.83	-2.34	+9.9%/-17.1%	±4.5%	+2.8%/-3.1%	±1.2%
20	-3.86	-3.00	-2.50	+10.0%/-17.4%	±4.6%	+3.0%/-2.6%	±1.1%
22	-4.08	-3.22	-2.69	+10.7%/-17.4%	±4.5%	+3.8%/-2.8%	±1.3%
24	-4.37	-3.48	-2.85	+12.9%/-17.8%	±4.7%	+2.6%/-2.5%	±1.2%
26	-4.48	-3.64	-3.12	+10.5%/-17.0%	±4.7%	+2.7%/-2.9%	±1.1%
28	-4.69	-3.82	-3.28	+10.8%/-17.7%	±4.8%	+3.7%/-4.4%	±2.3%
30	-4.94	-4.02	-3.49	+10.7%/-18.5%	±4.9%	+2.5%/-3.2%	±1.2%
32	-5.23	-4.28	-3.72	+11.3%/-19.1%	±5.1%	+3.7%/-2.5%	±1.2%
34	-5.56	-4.57	-4.00	+11.4%/-20.0%	±5.2%	+3.4%/-3.1%	±1.1%
36	-5.51	-4.56	-4.01	+11.2%/-19.1%	±5.1%	+2.6%/-2.9%	±1.1%
38	-5.93	-4.92	-4.30	+12.4%/-20.3%	±5.5%	+3.5%/-2.5%	±1.2%
40	-5.93	-4.92	-4.39	+10.8%/-20.4%	±5.5%	+2.6%/-2.5%	±1.2%
42	-6.41	-5.37	-4.71	+13.2%/-21.1%	±5.8%	+2.8%/-2.6%	±1.1%
44	-6.39	-5.36	-4.82	+11.0%/-20.8%	±5.8%	+2.9%/-2.6%	±1.2%
46	-7.02	-5.93	-5.31	+12.6%/-22.1%	±6.1%	+3.5%/-3.6%	±1.3%
48	-7.06	-5.93	-5.30	+12.6%/-22.8%	±6.0%	+3.0%/-2.6%	±1.3%
50	-7.02	-5.93	-5.25	+13.6%/-22.2%	±6.1%	+2.8%/-3.2%	±1.4%

Table A5.2 The Percentage Variations of the S_j to dX_i/dt Characteristic (V_{Tij}=2.00V). All percentages are taken relative to the mean response for V_{Tij}=5.00V and S_j=50%.

S_j (%)	V_{Tij} (V)	dX_i/dt (V/ms)				
		Min	Mean	Max	Variation	Std Dev
0	1.99	-1.64	-1.04	-0.56	+9.7%/-12.1%	±3.9%
0	5.00	0.12	0.80	1.37	+11.5%/-13.7%	±5.0%
5	1.99	-2.05	-1.51	-0.94	+11.4%/-11.0%	±3.9%
5	5.00	0.55	1.22	1.81	+11.8%/-13.7%	±5.2%
10	1.99	-2.55	-1.98	-1.40	+11.8%/-11.3%	±3.9%
10	5.00	0.96	1.64	2.24	+12.0%/-13.7%	±5.2%
15	1.99	-3.00	-2.49	-1.93	+11.3%/-10.2%	±4.0%
15	5.00	1.36	2.08	2.78	+13.9%/-14.6%	±5.3%
20	1.99	-3.47	-2.94	-2.43	+10.3%/-10.7%	±4.1%
20	5.00	1.78	2.46	3.16	+14.2%/-13.6%	±5.3%
25	1.99	-3.89	-3.43	-2.92	+10.4%/-9.2%	±4.2%
25	5.00	2.19	2.90	3.56	+13.3%/-14.4%	±5.7%
30	1.99	-4.47	-3.97	-3.44	+10.7%/-10.2%	±4.5%
30	5.00	2.63	3.39	3.98	+11.9%/-15.4%	±5.1%
35	1.99	-5.06	-4.51	-3.94	+11.3%/-11.1%	±4.8%
35	5.00	2.99	3.83	4.47	+12.9%/-16.8%	±5.4%
40	1.99	-5.45	-4.86	-4.28	+11.6%/-11.9%	±4.9%
40	5.00	3.25	4.12	4.88	+15.3%/-17.6%	±5.6%
45	1.99	-6.57	-5.87	-5.23	+12.8%/-14.2%	±5.5%
45	5.00	4.10	4.93	5.64	+14.3%/-16.8%	±6.1%
50	1.99	-6.56	-5.87	-5.19	+13.6%/-13.9%	±5.5%
50	5.00	4.16	4.94	5.66	+14.4%/-15.8%	±5.8%

Table A5.3 The Deviations from the Mean of the dX_i/dt Measurements for the Synaptic Multiplication Characteristic. All percentages are taken relative to the mean response for V_{Tij}=5.00V and S_j=50%.

S_j (%)	V_{Tij}	dX_i/dt (V/ms)	
		Variation	Standard Deviation
0	1.99	+3.2%/-2.7%	±1.8%
0	5.00	+2.5%/-3.8%	±1.5%
5	1.99	+2.5%/-2.2%	±1.2%
5	5.00	+4.2%/-3.4%	±1.7%
10	1.99	+2.5%/-2.3%	±1.2%
10	5.00	+3.5%/-4.4%	±1.8%
15	1.99	+3.3%/-3.8%	±2.0%
15	5.00	+3.0%/-4.8%	±1.8%
20	1.99	+2.4%/-2.8%	±1.5%
20	5.00	+5.2%/-4.6%	±2.1%
25	1.99	+2.3%/-3.4%	±1.4%
25	5.00	+6.7%/-8.8%	±3.7%
30	1.99	+2.6%/-2.4%	±1.3%
30	5.00	+2.8%/-7.9%	±2.7%
35	1.99	+2.8%/-2.5%	±1.2%
35	5.00	+4.1%/-6.0%	±2.1%
40	1.99	+2.4%/-3.4%	±1.5%
40	5.00	+4.7%/-6.8%	±2.7%
45	1.99	+3.2%/-2.7%	±1.5%
45	5.00	+3.0%/-4.8%	±1.9%
50	1.99	+2.4%/-2.8%	±1.6%
50	5.00	+3.3%/-4.7%	±1.8%

Table A5.4 The Accuracy of the Measurements of the S_j to dX_i/dt Characteristic. All percentages are taken relative to the mean response for V_{Tij}=5.00V and S_j=50%.

Appendix 6

Static Measurements of Process Variance for EPSILON 30120PI

V_{Tij} (V)	$V_{outi} - V_{0Z}$ (V)				
	HSPICE L2-Typ	Min	Mean	Max	Variation
0.00	-1.09	-1.17	-1.138	-1.11	+4.6%/-5.3%
0.25	-1.09	-1.17	-1.138	-1.11	+4.6%/-5.3%
0.50	-1.09	-1.17	-1.138	-1.11	+4.6%/-5.3%
0.75	-1.09	-1.17	-1.138	-1.11	+4.6%/-5.3%
1.00	-1.09	-1.17	-1.138	-1.11	+4.6%/-5.3%
1.25	-1.09	-1.17	-1.138	-1.11	+4.6%/-5.3%
1.50	-1.09	-1.17	-1.137	-1.11	+4.4%/-5.4%
1.75	-1.08	-1.13	-1.097	-1.07	+4.4%/-5.4%
2.00	-1.04	-1.02	-0.990	-0.96	+4.9%/-4.9%
2.25	-0.88	-0.87	-0.842	-0.82	+3.6%/-4.6%
2.50	-0.73	-0.71	-0.693	-0.67	+3.8%/-2.8%
2.75	-0.58	-0.57	-0.550	-0.52	+4.9%/-3.3%
3.00	-0.43	-0.44	-0.415	-0.39	+4.1%/-4.1%
3.25	-0.28	-0.30	-0.273	-0.25	+3.8%/-4.4%
3.50	-0.14	-0.17	-0.138	-0.11	+4.6%/-5.3%
3.75	0.00	-0.04	-0.007	0.02	+4.4%/-5.4%
4.00	0.14	0.09	0.123	0.15	+4.4%/-5.4%
4.25	0.28	0.22	0.252	0.27	+3.0%/-5.3%
4.50	0.42	0.33	0.377	0.40	+3.8%/-7.7%
4.75	0.56	0.45	0.497	0.52	+3.8%/-7.7%
5.00	0.69	0.56	0.608	0.64	+5.3%/-7.9%

Table A6.1 The Percentage Variations of the V_{Tij} to $V_{outi} - V_{0Z}$ Characteristic. All percentages are relative to the mean response for V_{Tij}=5.0V (0.608V).

V_{Tij} (V)	$V_{outi} - V_{0Z}$ (V)				
	HSPICE L2-Typ	Min	Mean	Max	Variation
0.00	0.03	0.01	0.042	0.07	+4.6%/-5.3%
0.25	0.03	0.01	0.042	0.07	+4.6%/-5.3%
0.50	0.03	0.01	0.042	0.07	+4.6%/-5.3%
0.75	0.03	0.01	0.042	0.07	+4.6%/-5.3%
1.00	0.03	0.01	0.042	0.07	+4.6%/-5.3%
1.25	0.03	0.01	0.042	0.07	+4.6%/-5.3%
1.50	0.03	0.01	0.042	0.07	+4.6%/-5.3%
1.75	0.03	0.01	0.042	0.07	+4.6%/-5.3%
2.00	0.03	0.00	0.035	0.06	+4.1%/-5.8%
2.25	0.03	-0.01	0.028	0.05	+3.6%/-6.2%
2.50	0.02	-0.02	0.018	0.04	+3.6%/-6.2%
2.75	0.02	-0.02	0.013	0.04	+4.4%/-5.4%
3.00	0.01	-0.03	0.003	0.03	+4.4%/-5.4%
3.25	0.01	-0.04	-0.002	0.02	+3.6%/-6.2%
3.50	0	-0.04	-0.005	0.02	+4.1%/-5.8%
3.75	0	-0.04	-0.007	0.02	+4.4%/-5.4%
4.00	0	-0.05	-0.012	0.01	+3.6%/-6.3%
4.25	0.01	-0.05	-0.015	0.01	+4.1%/-5.8%
4.50	-0.01	-0.05	-0.017	0.01	+4.4%/-5.4%
4.75	-0.02	-0.05	-0.017	0.01	+4.4%/-5.4%
5.00	-0.02	-0.05	-0.018	0.00	+3.0%/-5.3%

Table A6.2 The Percentage Variation in the V_{outiZ} as a Function of V_{Tij} . All percentages are relative to the mean response for V_{Tij}=5.0V (0.608V).

Appendix 7

Dynamic Measurements of Process Variance for EPSILON 30120PI

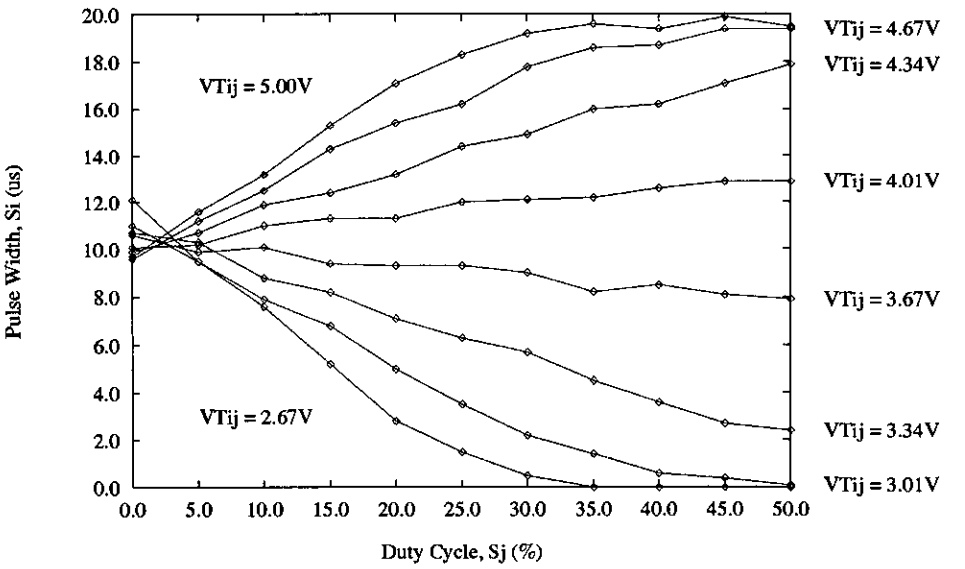


Figure A7.1 PFM Input Mode Multiplication Characteristic.

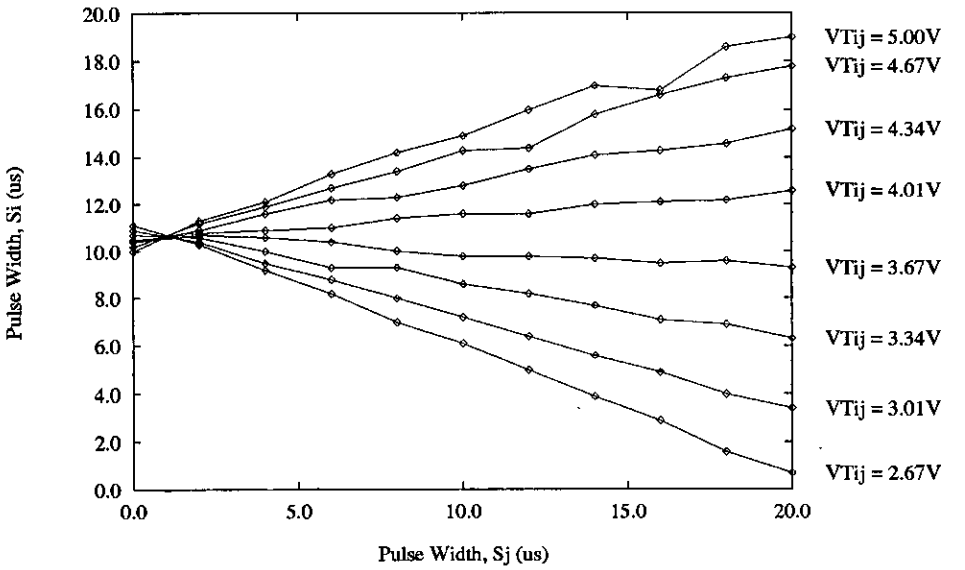


Figure A7.2 PWM Input Mode Multiplication Characteristic.

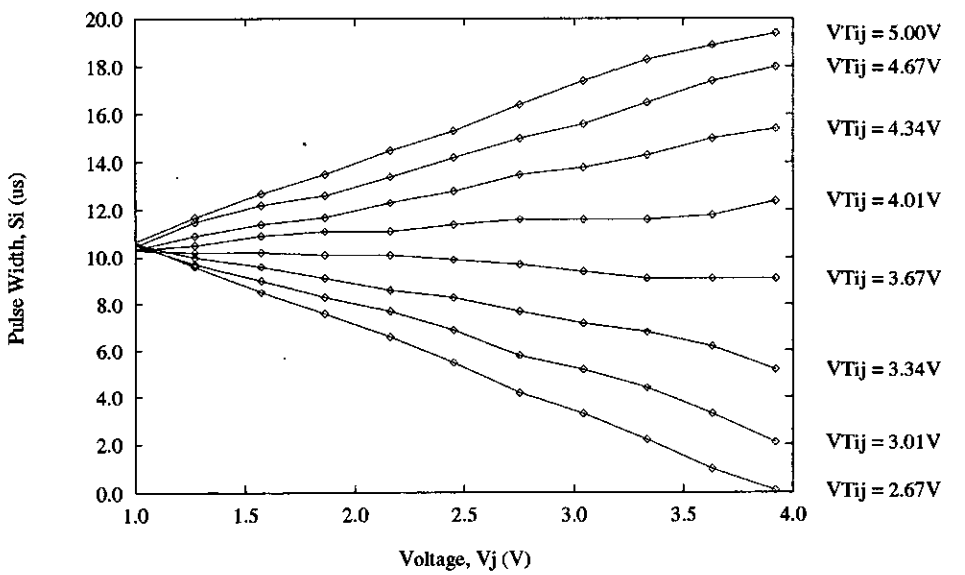


Figure A7.3 Analogue Input Mode Multiplication Characteristic.

Appendix 8

Dynamic Measurements of Process Variance

for EPSILON 30120PI

V_{Tij} (V)	S_i (μs)				
	Min	Mean	Max	Variation	Std Dev
5.00	12.20	13.40	14.70	+38.2%/-35.3%	±12.0%
4.92	12.10	13.20	14.50	+38.2%/-32.4%	±12.3%
4.84	11.80	13.00	14.10	+32.4%/-35.3%	±11.8%
4.76	11.60	12.70	13.90	+35.3%/-32.4%	±11.6%
4.67	11.50	12.50	13.60	+32.4%/-29.4%	±11.2%
4.59	11.30	12.30	13.40	+32.4%/-29.4%	±10.9%
4.51	11.10	12.10	13.30	+35.3%/-29.4%	±11.0%
4.42	10.90	11.80	12.80	+29.4%/-26.5%	±10.1%
4.34	10.60	11.60	12.60	+29.4%/-29.4%	±9.9%
4.26	10.40	11.40	12.40	+29.4%/-29.4%	±9.5%
4.17	10.20	11.10	12.20	+32.4%/-26.5%	±9.7%
4.09	9.90	10.90	12.00	+32.4%/-29.4%	±9.6%
4.01	9.70	10.70	11.60	+26.5%/-29.4%	±9.8%
3.92	9.50	10.40	11.40	+29.4%/-26.5%	±9.9%
3.84	9.30	10.20	11.10	+26.5%/-26.5%	±9.7%
3.76	9.10	9.90	10.90	+29.4%/-23.5%	±9.3%
3.67	8.80	9.70	10.60	+26.5%/-26.5%	±8.5%
3.59	8.60	9.50	10.50	+29.4%/-26.5%	±8.1%
3.51	8.50	9.20	10.00	+23.5%/-20.6%	±7.9%
3.42	8.20	9.00	9.80	+23.5%/-23.5%	±7.9%
3.34	8.00	8.80	9.60	+23.5%/-23.5%	±7.8%
3.26	7.80	8.60	9.50	+26.5%/-23.5%	±7.8%
3.17	7.60	8.40	9.10	+20.6%/-23.5%	±7.6%
3.09	7.30	8.20	9.00	+23.5%/-26.5%	±7.8%
3.01	6.90	7.90	8.80	+26.5%/-29.4%	±8.4%
2.92	6.70	7.70	8.40	+20.6%/-29.4%	±8.3%
2.84	6.40	7.50	8.30	+23.5%/-32.4%	±9.1%
2.76	6.30	7.30	8.10	+23.5%/-29.4%	±8.6%
2.67	6.00	7.10	7.90	+23.5%/-32.4%	±9.0%
2.59	6.10	6.90	7.60	+20.6%/-23.5%	±8.6%
2.51	5.80	6.80	7.50	+20.6%/-29.4%	±8.7%

Table A8.1 The Percentage Variations of the V_{Tij} to S_i Characteristic (S_j=75.4V). All percentages are relative to the mean response for V_{Tij}=5.0V and S_j= 75.4μs (13.4μs-10.0μs).

V_{Tij} (V)	S_i (μs)				
	Min	Mean	Max	Variation	Std Dev
0.00	8.80	9.30	10.10	+23.5%/-14.7%	±5.0%
4.00	9.00	9.50	10.40	+26.5%/-14.7%	±5.6%
8.00	9.20	9.70	10.90	+35.3%/-14.7%	±6.7%
12.00	9.40	9.90	10.90	+29.4%/-14.7%	±6.3%
16.00	9.60	10.20	11.50	+38.2%/-17.6%	±7.3%
20.00	9.80	10.40	11.20	+23.5%/-17.6%	±6.0%
24.00	9.90	10.70	11.60	+26.5%/-23.5%	±6.6%
28.00	10.10	10.90	12.00	+32.4%/-23.5%	±7.1%
32.00	10.40	11.10	11.90	+23.5%/-20.6%	±7.1%
36.00	10.60	11.30	12.30	+29.4%/-20.6%	±7.8%
40.00	10.70	11.50	12.60	+32.4%/-23.5%	±8.2%
44.00	11.00	11.80	12.80	+29.4%/-23.5%	±8.4%
48.00	11.10	12.00	12.90	+26.5%/-26.5%	±8.9%
51.40	11.20	12.20	13.20	+29.4%/-29.4%	±9.3%
53.40	11.30	12.30	13.20	+26.5%/-29.4%	±9.2%
55.40	11.40	12.40	13.40	+29.4%/-29.4%	±9.6%
57.40	11.50	12.50	13.80	+38.2%/-29.4%	±10.5%
59.40	11.50	12.60	14.00	+41.2%/-32.4%	±10.8%
61.40	11.70	12.70	13.70	+29.4%/-29.4%	±10.5%
63.40	11.70	12.80	13.90	+32.4%/-32.4%	±10.7%
65.40	11.70	12.90	14.00	+32.4%/-35.3%	±10.7%
67.40	12.00	13.00	14.20	+35.3%/-29.4%	±10.7%
69.40	11.90	13.10	14.40	+38.2%/-35.3%	±11.6%
71.40	11.90	13.20	14.50	+38.2%/-38.2%	±11.9%
73.40	12.10	13.30	14.50	+35.3%/-35.3%	±11.5%
75.40	12.20	13.40	14.70	+38.2%/-35.3%	±12.0%

Table A8.2 The Percentage Variations of the S_j to S_i Characteristic (V_{Tij}=5.00V). All percentages are relative to the mean response for V_{Tij}=5.0V and S_j= 75.4μs (13.4μs-10.0μs).

S_j (μs)	V_{Tij} (V)	S_i (μs)				
		Min	Mean	Max	Variation	Std Dev
0.0	5.00	8.70	9.20	10.40	+35.3%/-14.7%	±6.3%
50.0	5.00	10.90	12.00	13.80	+52.9%/-32.4%	±10.9%
75.4	5.00	12.10	13.40	14.50	+32.4%/-38.2%	±12.2%
0.0	4.76	8.70	9.30	9.90	+17.6%/-17.6%	±5.3%
50.0	4.76	10.60	11.60	13.10	+44.1%/-29.4%	±11.8%
75.4	4.76	11.50	12.70	14.50	+52.9%/-35.3%	±12.2%
0.0	4.42	9.00	9.50	10.10	+17.6%/-14.7%	±5.3%
50.0	4.42	10.20	11.10	12.30	+35.3%/-26.5%	±9.5%
75.4	4.42	10.50	11.80	12.80	+29.4%/-38.2%	±10.7%
0.0	4.09	9.20	9.70	10.20	+14.7%/-14.7%	±5.0%
50.0	4.09	9.50	10.50	11.30	+23.5%/-29.4%	±9.0%
75.4	4.09	9.90	10.90	11.80	+26.5%/-29.4%	±9.7%
0.0	3.76	9.40	9.90	10.40	+14.7%/-14.7%	±4.7%
50.0	3.76	8.90	9.90	10.80	+26.5%/-29.4%	±7.8%
75.4	3.76	8.90	9.90	10.80	+26.5%/-29.4%	±9.9%
0.0	3.42	9.60	10.30	10.70	+11.8%/-20.6%	±5.6%
50.0	3.42	8.60	9.40	10.10	+20.6%/-23.5%	±6.1%
75.4	3.42	8.10	9.00	9.80	+23.5%/-26.5%	±8.2%
0.0	3.09	10.10	10.70	11.10	+11.8%/-17.6%	±5.3%
50.0	3.09	8.00	8.90	9.40	+14.7%/-26.5%	±7.3%
75.4	3.09	7.10	8.10	8.80	+20.6%/-29.4%	±8.8%
0.0	2.76	10.70	11.30	11.70	+11.8%/-17.6%	±5.4%
50.0	2.76	7.60	8.50	9.20	+20.6%/-26.5%	±8.2%
75.4	2.76	6.30	7.30	8.10	+23.5%/-29.4%	±8.6%

Table A8.3 The Deviations from the Mean of the S_i Measurements for the Synaptic Multiplication Characteristic. All percentages are relative to the mean response for V_{Tij}=5.0V and S_j= 75.4μs (13.4μs-10.0μs).

Appendix 9

Measurement Accuracy for EPSILON 30120PI Results

S_j (μs)	V_{Tij} (V)	S_i (μs)				
		Min	Mean	Max	Variation	Std Dev
75.4	5.00	13.00	13.49	13.80	+9.0%/-14.5%	±6.0%
75.4	4.92	12.60	13.46	17.00	+104.1%/-25.3%	±22.4%
75.4	4.84	12.60	13.13	13.80	+19.7%/-15.6%	±7.8%
75.4	4.76	12.20	13.03	16.30	+96.1%/-24.5%	±19.2%
75.4	4.67	12.30	12.77	15.80	+89.0%/-13.9%	±17.3%
75.4	4.59	11.90	12.47	13.00	+15.5%/-16.8%	±6.3%
75.4	4.51	11.80	12.26	12.70	+13.0%/-13.5%	±6.9%
75.4	4.42	9.50	11.98	12.50	+15.4%/-72.8%	±12.3%
75.4	4.34	11.40	11.88	13.80	+56.4%/-14.2%	±13.2%
75.4	4.26	9.90	11.60	13.70	+61.8%/-50.0%	±12.6%
75.4	4.17	10.80	11.41	12.90	+43.7%/-18.1%	±8.8%
75.4	4.09	10.70	11.21	12.20	+29.1%/-15.0%	±8.3%
75.4	4.01	9.00	10.89	11.40	+15.1%/-55.5%	±10.5%
75.4	3.92	10.30	10.71	11.40	+20.4%/-12.0%	±6.6%
75.4	3.84	9.80	10.49	10.90	+12.2%/-20.2%	±7.3%
75.4	3.76	9.70	10.22	10.80	+16.9%/-15.4%	±8.7%
75.4	3.67	9.10	9.95	10.50	+16.2%/-25.0%	±9.0%
75.4	3.59	9.40	9.74	10.20	+13.5%/-10.0%	±7.1%
75.4	3.51	8.50	9.50	10.00	+14.8%/-29.3%	±7.2%
75.4	3.42	7.80	9.26	9.80	+15.9%/-42.9%	±9.7%
75.4	3.34	8.60	9.08	9.50	+12.2%/-14.2%	±6.7%
75.4	3.26	7.50	8.88	9.40	+15.2%/-40.6%	±8.8%
75.4	3.17	7.20	8.64	9.10	+13.4%/-42.5%	±8.7%
75.4	3.09	8.20	8.52	8.90	+11.2%/-9.4%	±6.1%
75.4	3.01	7.70	8.32	8.80	+14.2%/-18.1%	±6.3%
75.4	2.92	4.60	7.99	8.50	+14.9%/-99.8%	±19.9%
75.4	2.84	4.60	7.89	8.40	+15.0%/-96.8%	±14.9%
75.4	2.76	7.40	7.75	8.20	+13.2%/-10.3%	±6.3%
75.4	2.67	6.60	7.58	8.00	+12.2%/-28.9%	±7.3%
75.4	2.59	3.30	7.35	7.90	+16.2%/-119.1%	±17.9%
75.4	2.51	7.00	7.35	7.70	+10.4%/-10.2%	±5.7%

Table A9.1 Percentage Measurement Variation in Samples for the V_{Tij} to S_i EPSILON Characteristic. All percentages are relative to the mean response for V_{Tij}=5.0V and S_j=75.4μs (Results from Chip 3, Row 1, Column 1).

S_j (μs)	V_{Tij} (V)	S_i (μs)				
		Min	Mean	Max	Variation	Std Dev
0.0	5.00	8.50	9.24	9.60	+10.6%/-21.8%	±6.6%
4.0	5.00	9.10	9.50	10.00	+14.6%/-11.8%	±5.4%
8.0	5.00	9.30	9.79	13.80	+118.0%/-14.4%	±23.6%
12.0	5.00	9.50	9.88	10.50	+18.2%/-11.2%	±6.9%
16.0	5.00	9.70	10.21	10.70	+14.4%/-15.0%	±8.5%
20.0	5.00	10.10	10.49	10.90	+11.9%/-11.6%	±5.7%
24.0	5.00	10.40	10.74	11.10	+10.6%/-9.9%	±5.0%
28.0	5.00	10.60	11.01	14.50	+102.6%/-12.1%	±15.9%
32.0	5.00	10.70	11.19	11.70	+14.9%/-14.5%	±7.4%
36.0	5.00	10.90	11.41	11.80	+11.5%/-15.0%	±7.4%
40.0	5.00	11.20	11.60	12.00	+11.9%/-11.6%	±6.0%
44.0	5.00	11.50	11.82	12.20	+11.2%/-9.4%	±5.8%
48.0	5.00	11.70	12.04	12.50	+13.5%/-10.0%	±6.0%
51.4	5.00	11.70	12.25	12.80	+16.1%/-16.3%	±6.7%
53.4	5.00	11.80	12.39	12.90	+15.0%/-17.4%	±7.2%
55.4	5.00	12.10	12.54	13.00	+13.5%/-13.0%	±6.9%
57.4	5.00	12.20	12.61	13.20	+17.4%/-12.1%	±6.9%
59.4	5.00	12.20	12.69	13.10	+12.1%/-14.4%	±6.4%
61.4	5.00	12.30	12.88	16.60	+109.4%/-17.1%	±17.1%
63.4	5.00	12.50	12.98	16.20	+94.8%/-14.1%	±15.2%
65.4	5.00	12.60	13.05	13.60	+16.3%/-13.1%	±6.8%
67.4	5.00	12.70	13.25	17.20	+116.2%/-16.2%	±17.6%
69.4	5.00	12.70	13.33	17.40	+119.6%/-18.6%	±18.5%
71.4	5.00	13.00	13.39	13.80	+11.9%/-11.6%	±6.1%
73.4	5.00	13.00	13.48	13.90	+12.4%/-14.1%	±6.8%
75.4	5.00	13.30	13.75	17.90	+122.2%/-13.1%	±24.4%

Table A9.2 Percentage Measurement Variation in Samples for the S_{Tij} to S_i EPSILON Characteristic. All percentages are relative to the mean response for V_{Tij}=5.0V and S_j=75.4μs (Results from Chip 3, Row 1, Column 1).

Appendix 10

Results of Software/Hardware Comparison

Neuron	City 5			City 9		
	Software	EPSILON	Variation	Software	EPSILON	Variation
0	0.302	-	-	0.250	-	-
1	0.302	0.294	-2.4%	0.251	0.259	2.4%
2	0.249	0.237	-3.6%	0.313	0.329	4.8%
3	0.168	0.127	-12.3%	0.333	0.355	6.6%
4	0.167	0.127	-12.0%	0.333	0.337	1.2%
5	0.168	0.114	-16.2%	0.333	0.333	0.0%
6	0.249	0.241	-2.4%	0.313	0.333	6.0%
7	0.302	0.294	-2.4%	0.251	0.263	3.6%
8	0.302	0.289	-3.9%	0.250	0.259	2.7%
9	0.302	0.294	-2.4%	0.248	0.259	3.3%
10	0.289	0.267	-6.6%	0.171	0.158	-3.9%
11	0.287	0.280	-2.1%	0.167	0.158	-2.7%
12	0.288	0.267	-6.3%	0.167	0.153	-4.2%
13	0.306	0.298	-2.4%	0.130	0.123	-2.1%
14	0.312	0.285	-8.1%	0.084	0.079	-1.5%
15	0.312	-	-	0.084	-	-
16	0.312	0.285	-8.1%	0.081	0.070	-3.3%
17	0.290	0.272	-5.4%	0.004	0.009	1.5%
18	0.289	0.254	-10.5%	0.000	0.004	1.2%
19	0.290	0.250	-12.0%	0.004	-0.004	-2.4%
20	0.312	0.294	-5.4%	0.081	0.079	-0.6%
21	0.312	0.302	-3.0%	0.084	0.088	1.2%
22	0.313	0.302	-3.3%	0.088	0.088	0.0%
23	0.333	0.333	0.0%	0.164	0.175	3.3%
24	0.333	0.333	0.0%	0.167	0.184	5.1%
25	0.333	0.298	-10.5%	0.168	0.158	-3.0%
26	0.291	0.285	-1.8%	0.168	0.158	-3.0%
27	0.287	0.272	-4.5%	0.167	0.158	-2.7%
28	0.289	0.267	-6.6%	0.171	0.153	-5.4%
29	0.302	0.289	-3.9%	0.248	0.254	1.8%

Table A10.1 The Percentage Variation between the Responses of EPSILON and Software, for the City 5 and City 9 Vectors applied to the Kohonen Neural Network for the Example 9 City TSP. All percentages are relative to 0.333.

Appendix 11

A Continuous Time Analogue Synaptic Multiplier

The circuits outlined in this appendix combine the process invariance of the distributed feedback synapse (Chapters 4-6) with the 2 transistor 4 quadrant transconductance multiplier (Section 3.3.3) to produce a continuous time, process invariant, fully integrated, fast analogue multiplier. The multiplication is now purely analogue as the neural state is represented by an analogue voltage, V_j, rather than as a pulse frequency or a pulse width.

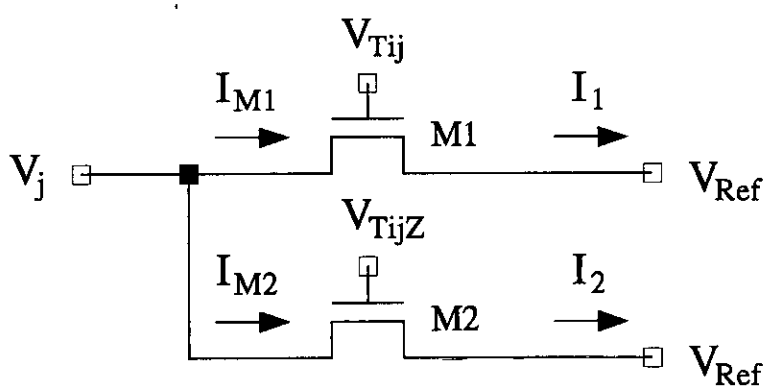


Figure A11.1 2 Transistor MOS Transconductor.

A11.1 The Design of a 2-Quadrant Analogue Multiplier

As Equation A11.1 shows, the 2 transistor transconductor (figure A11.1) possesses an accurate 4 quadrant multiplication characteristic [1-6].

$$I_1 - I_2 = \beta\left[(V_{Tij} - V_{TijZ})(V_j - V_{Ref})\right] \qquad \text{(A11.1)}$$

The supply of the reference voltages for the transistors' gates is straightforward but to maintain the correct voltages at the drain and source terminals requires buffers which can sink and source current. The Denyer/Mavor implementation [3] uses a simple MOS source follower buffer to set the drain voltage with an operational amplifier with resistive feedback maintaining the virtual earth, V_{Ref}, at the source as well as summing the current from a column of these multipliers. In this setup both the output voltage of the source

follower buffer **and** the output current of the transconductor are subject to the effects of process variations. This is due to the source follower output being dependent on the threshold voltage and multiplier output current depending on the value of β. There are also cascadability problems similar to those of the 3 transistor pulse stream synapse (Section 4.2.1). The reported variation of these circuits implemented on a properly setup analogue MOS process is less than 1%. From the results presented in Chapters 5 and 6, ES2's 2μm or 1.5μm digital processes cannot be expected to be as uniform. Distributed feedback was used to combat the effects of process variations.

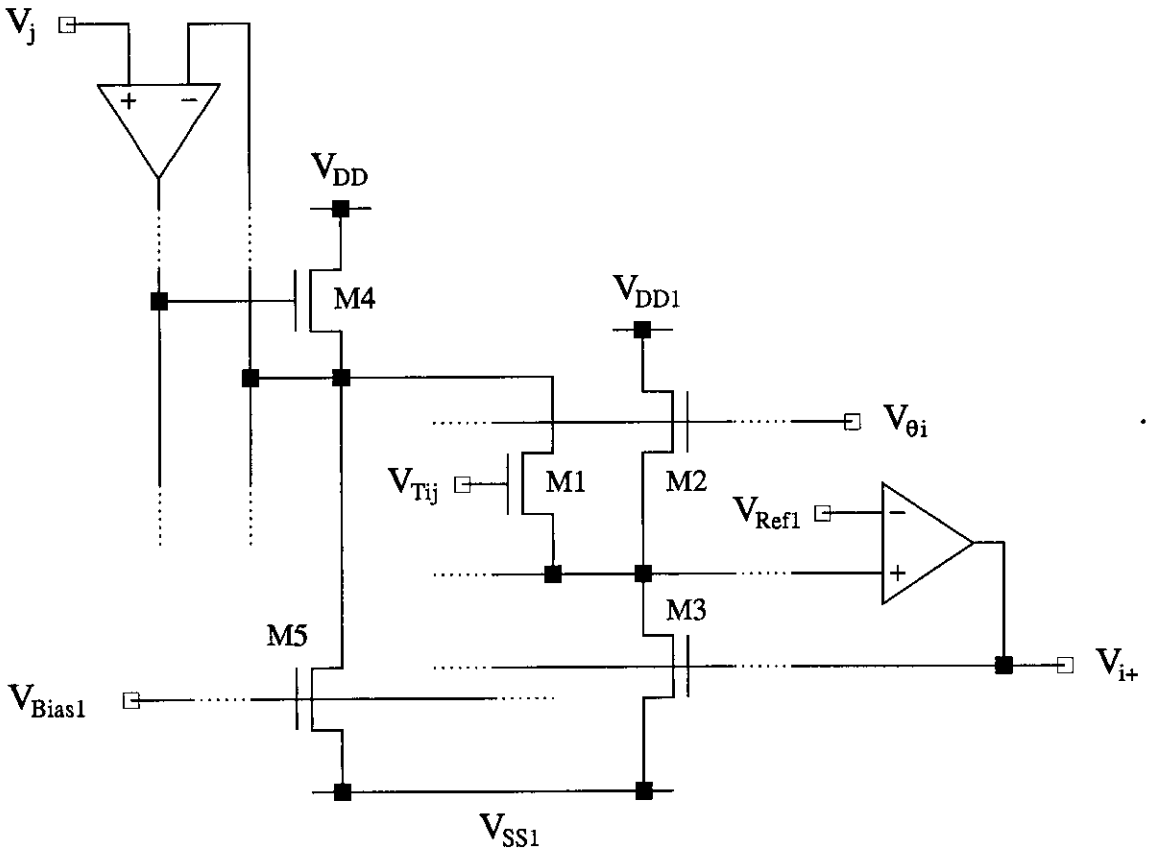


Figure A11.2 A Completely Analogue Synapse.

The resultant feedback circuitry for a single multiplier transistor (M1) is shown in Figure A11.2. Two of these circuits form the desired multiplier. As in the distributed feedback synapse the output is a voltage rather than a current. The final result is represented as a voltage difference as opposed to a current difference. The feedback loop setting V_j holds the drain of transistor M1 at the desired V_j by controlling the gate voltage of transistor M4 which is in its linear operating region. Transistor M5 is present to ensure that the drain of M1 is always actively driven towards the desired V_j even if M1 sinks no current (V_j=V_{Ref}). This permanent current flow in the M4/M5 leg ensures a fast response

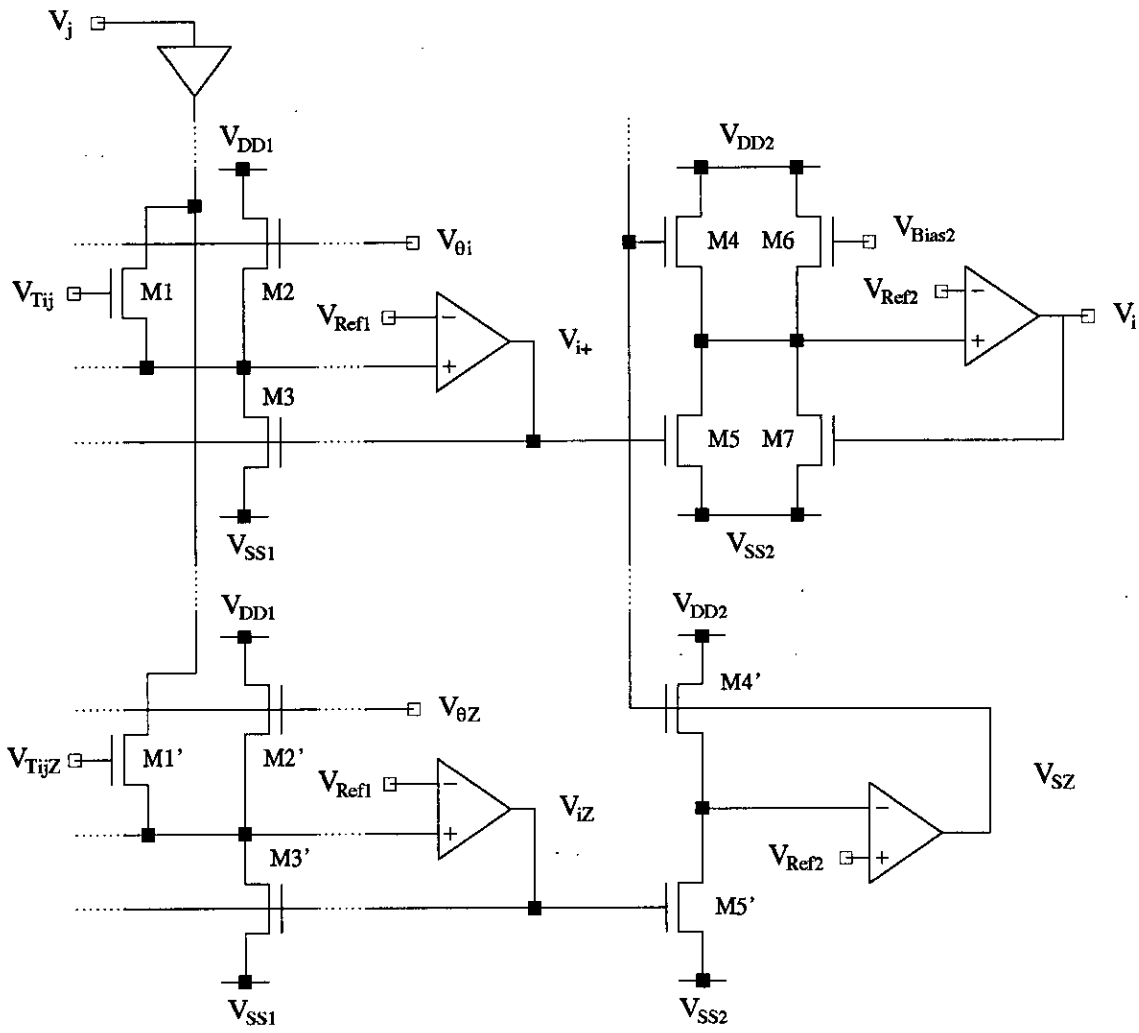


Figure A11.3 The Components of Continuous Time Synaptic Multiplication System.

to a change in V_j. Transistors M2 and M3 form a buffer stage identical to that used in the distributed feedback synapse, thus forming a current to voltage converter. This makes the overall voltage to voltage transfer function theoretically process invariant.

To prove this result it is necessary to analyse the circuit in Figure A11.2 in more detail. As the combination of transistors M4, M5 and the input operational amplifier accurately maintain the value of V_j, they have no effect of the circuit's multiplication characteristic. The currents for the remaining transistors M1, M2 and M3 which are in their linear regions of operation, are as follows

$$I_{M1} = \beta_1 \left[(V_{Tij} - V_{Ref} - V_{T1})(V_j - V_{Ref}) - \frac{(V_j - V_{Ref})^2}{2} \right] \qquad \text{(A11.2)}$$

$$I_{M2} = \beta_2 \left[(V_{\theta i} - V_{Ref} - V_{T2})V_{Ref} - \frac{V_{Ref}^2}{2} \right] \quad \text{(A11.3)}$$

$$I_{M3} = \beta_3 \left[(V_{i+} - V_{T3})V_{Ref} - \frac{V_{Ref}^2}{2} \right] \quad \text{(A11.4)}$$

Applying Kirchhoff's Current Law to the common node between M1, M2 and M3 yields

$$I_{M1} + I_{M2} - I_{M3} = 0 \quad \text{(A11.5)}$$

Substituting Equations A11.2, A11.3 and A11.4 into Equation A11.5 and solving for V_{i+} now gives

$$V_{i+} = \left\{ \begin{array}{l} \frac{1}{V_{Ref}} \frac{\beta_1}{\beta_3} \left[(V_{Tij} - V_{Ref} - V_{T1})(V_j - V_{Ref}) - \frac{(V_j - V_{Ref})^2}{2} \right] \\ \quad + \frac{\beta_2}{\beta_3} \left[V_{\theta i} - V_{Ref} - V_{T2} - \frac{V_{Ref}}{2} \right] + V_{T3} + \frac{V_{Ref}}{2} \end{array} \right\} \quad \text{(A11.6)}$$

The equation for the the second half of the multiplier circuit is obtained by substituting V_{TijZ} and $V_{\theta iZ}$ for V_{Tij} and $V_{\theta i}$ respectively.

$$V_{iZ} = \left\{ \begin{array}{l} \frac{1}{V_{Ref}} \frac{\beta_1}{\beta_3} \left[(V_{TijZ} - V_{Ref} - V_{T1})(V_j - V_{Ref}) - \frac{(V_j - V_{Ref})^2}{2} \right] \\ \quad + \frac{\beta_2}{\beta_3} \left[V_{\theta iZ} - V_{Ref} - V_{T2} - \frac{V_{Ref}}{2} \right] + V_{T3} + \frac{V_{Ref}}{2} \end{array} \right\} \quad \text{(A11.7)}$$

Subtracting Equation A11.7 from Equation A11.6 cancels out the non-linear terms leaving

$$V_{i+} - V_{iZ} = \frac{1}{V_{Ref}} \frac{\beta_1}{\beta_3} \left[(V_{Tij} - V_{TijZ})(V_j - V_{Ref}) \right] + \frac{\beta_2}{\beta_3} (V_{\theta i} - V_{\theta iZ}) \quad \text{(A11.8)}$$

This equation can now be extended to a column of N multipliers

$$V_{i+} - V_{iZ} = \frac{1}{N} \frac{1}{V_{Ref}} \frac{\beta_1}{\beta_3} \sum_{j=0}^{N-1} \left[(V_{Tij} - V_{TijZ})(V_j - V_{Ref}) \right] + \frac{\beta_2}{\beta_3} (V_{\theta i} - V_{\theta iZ}) \quad \text{(A11.9)}$$

This shows that the voltage difference, $(V_{i+} - V_{iZ})$, is proportional to the sum of $V_{Tij}V_j$ multiplications. Due to the presence of β ratios in the equation, the response is process invariant to a first order. The second term in the equation is a bonus in that it allows each synaptic column to have a variable offset which can be used either to compensate for process variations or to implement the variable biases required by neurons in

the MLP and Hopfield/Tank neural networks.

The circuitry to perform the subtraction necessary to obtain the final result is shown on the left-hand-side of Figure A11.3. The equations for these circuits are

$$V_{SZ} = V_{iZ} + V_{Ref2} + \Delta V_T \tag{A11.10}$$

$$V_i = V_{i+} - V_{SZ} + V_{Ref} - \Delta V_T + V_{Bias} + V_{Ref} + \Delta V_T \tag{A11.11}$$

Thus combining

$$V_i = (V_{iZ} - V_{i+}) + V_{Bias} + V_{Ref} + \Delta V_T \tag{A11.12}$$

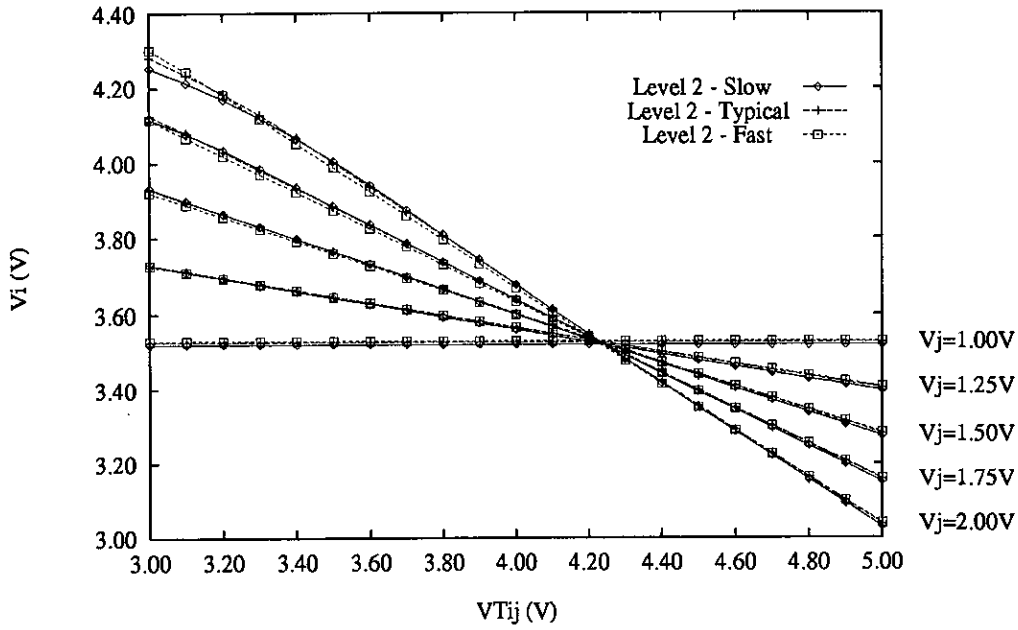


Figure A11.4 Multiplier Characteristic for a sweep of V_{Tij} from 3.0V to 5.0V and V_j from 1.0V to 2.0V (0.25V steps).

The subtraction is not quite what is required as the order of the coefficients is wrong. However, this is solved simply by using the lower half of the weight voltage range to represent excitatory weights, and the upper half to represent inhibitory weights.

A quick component count reveals that to carry out a single multiplication using this circuitry requires at least 5 operational amplifiers plus 2 multiplier cells and a subtraction cell. Thus the area and power required to achieve a single multiplication is excessive but these overheads are much smaller when an array of multipliers is considered. For a 10 by 10 synaptic array only 10 input operational amplifiers are required, one per V_j input. The nature of this vector-into-matrix multiplication means that only one column of zero

V_{Tij} (V)	V_j (V)	V_i (V)			Simulated Variance
		Slow	Typ	Fast	
3.00	1.00	3.522	3.529	3.531	+1.8%/+0.6%
3.50	1.00	3.522	3.529	3.531	+1.8%/+0.6%
4.00	1.00	3.522	3.529	3.531	+1.9%/+0.7%
4.50	1.00	3.521	3.529	3.531	+1.9%/+0.7%
5.00	1.00	3.521	3.528	3.531	+2.0%/+0.7%
3.00	1.50	3.933	3.930	3.922	-0.7%/-2.3%
3.50	1.50	3.766	3.766	3.760	+0.2%/-1.5%
4.00	1.50	3.601	3.604	3.601	+0.8%/-0.8%
4.50	1.50	3.438	3.442	3.442	+1.3%/-0.1%
5.00	1.50	3.275	3.283	3.285	+2.0%/+0.6%
3.00	2.00	4.253	4.282	4.302	+8.0%/+5.3%
3.50	2.00	4.007	4.003	3.989	-1.0%/-3.8%
4.00	2.00	3.679	3.678	3.670	-0.2%/-2.3%
4.50	2.00	3.352	3.356	3.353	+1.0%/-0.9%
5.00	2.00	3.029	3.037	3.039	+2.2%/+0.6%

Table A11.1 HSPICE Simulation Results for the Continuous Time System. Percentages are relative to voltage difference between the typical response to V_j=1.0V and V_j=2.0V for V_{Tij}-5.0V.

reference (V_{iZ}) multipliers is required per array. Finally a column of synapses needs just two operational amplifiers, one to calculate the sum of the multiplications and a second to subtract the zero reference to give the required result. So for a 10 by 10 array (100 multiplications) only 32 (10+2+20) operational amplifiers are required. As the number of operational amplifiers scales linearly with array size the continuous time synaptic multiplier is well suited to large 2 dimensional arrays eg 128 by 32.

In general for MLP's, the Hopfield/Tank and the Kohonen neural networks, the neural states are unipolar, 0 to 1.0. This means that full 4 quadrant multiplication is not necessary and 2 quadrant multiplication is perfectly adequate. Thus the setup of the continuous multiplier synapse was optimised for 2 quadrant multiplication.

The voltage ranges for V_{Tij} and V_j have to be carefully chosen to ensure that, for all combinations, transistors M1, M2 and M3 are always in their linear regions of operation. For this implementation $3.5V \leq V_{Tij} \leq 5.0V$ and $1.0V \leq V_j \leq 2.0V$. The HSPICE Level 2 simulation results shown in Figure A11.4 and Table A11.1, confirm both the linearity

and the process invariance of the proposed continuous time neural system over the chosen input ranges. Also the system is cascadable with the buffer stages formed by transistors M2, M3, M4 and M5 supplying the current required by the multiplier transistor M1 rather than the operational amplifier.

Bearing in mind the results for the distributed feedback synapse reported in Chapters 5 and 6, the simulated process variation of ±2% for the continuous time synapse is probably not a good indication of its performance on either the ECDM20 or the ECPD15/1 process. However with a good analogue process it would not be unreasonable to expect the effects of process variations to be less than 1% and thus comparable to the work of Mavor and Denyer [3].

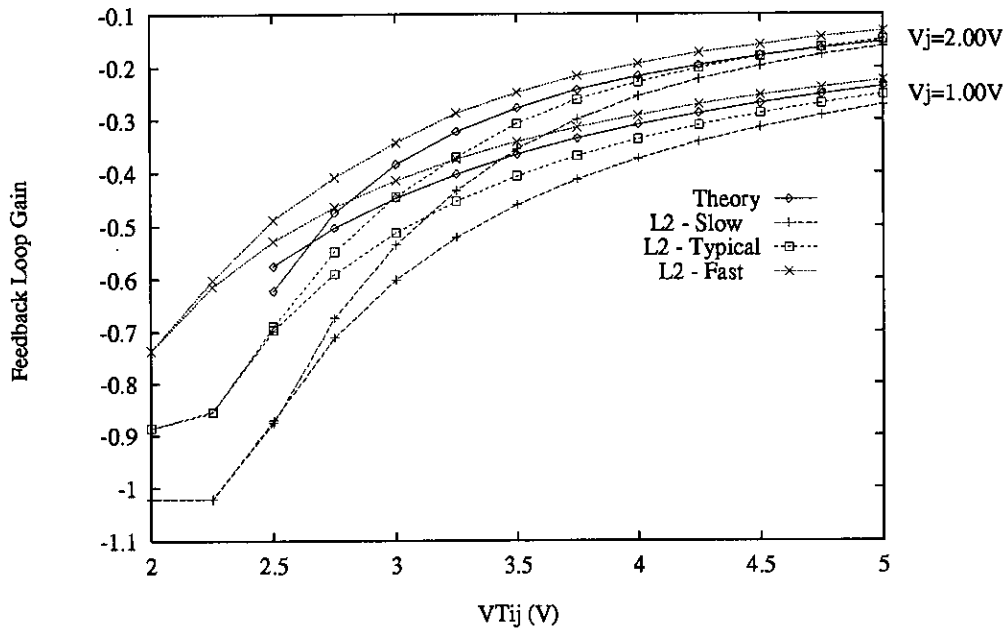


Figure A11.5 Feedback Loop Gains for a sweep of V_{Tij} from 3.0V to 5.0V and V_j from 1.0V and 2.0V.

A11.2 Stability Analysis and Operational Amplifier Design

To find the gain around the feedback loop an expression relating V_{Ref} to V_{outi} needs to be derived. The first step is to substitute Equations 5.2, 5.3, and 5.4 into Equation A11.5. and re-arrange in terms in powers of V_{Ref}.

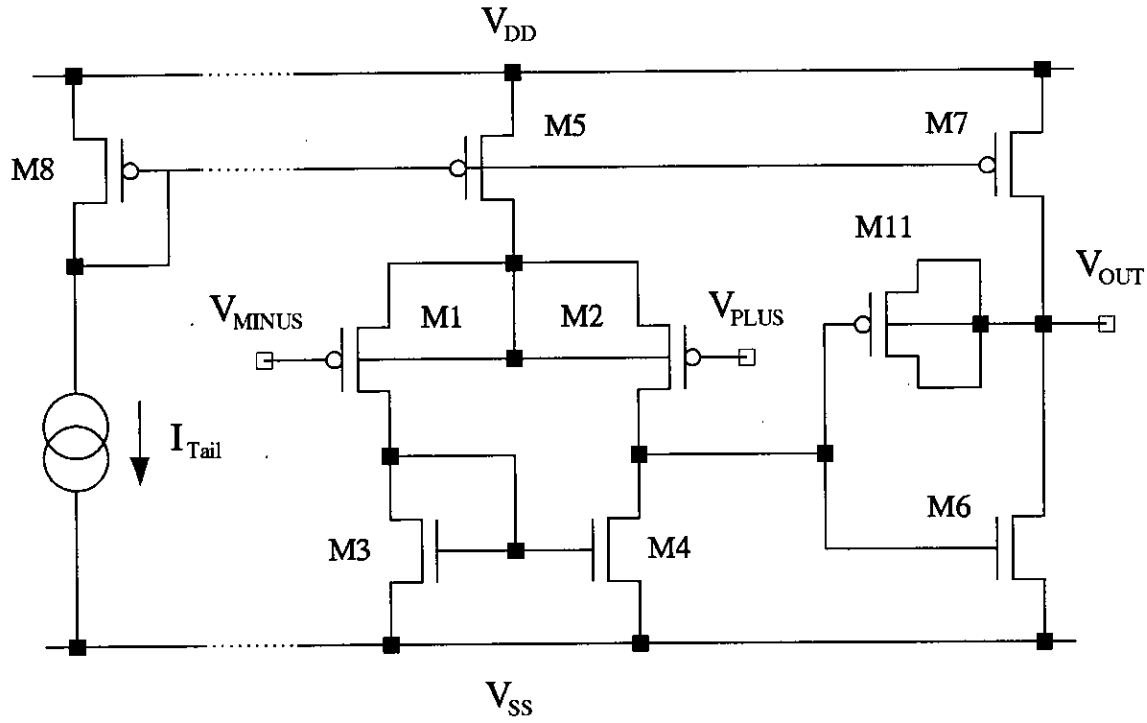


Figure A11.6 2-Stage Operational Amplifier with Controllable Tail Current.

$$\frac{1}{2}(\beta_1+\beta_2+\beta_3)V_{Ref}{}^2 \qquad \text{(A11.13)}$$

$$+\left[\beta_1(V_{T1}-V_{Tij})+\beta_2(V_{T2}-V_{\theta i})+\beta_3(V_{T3}-V_{i+})\right]V_{Ref}$$

$$+\beta_1((V_{Tij}-V_{T1}-\frac{V_j}{2})V_j)+\beta_2((V_{\theta i}-V_{T2}-\frac{V_{DD}}{2})V_{DD})=0$$

The feedback loop gain is then found by using the quadratic root equation (Equation 4.25) and differentiating the result with respect to V_{i+}

$$\frac{dV_{Ref}}{dV_{i+}}=\frac{1}{2a}\left[\beta_3\pm\frac{1}{2}\frac{-2b\beta_3}{\sqrt{b^2-4ac}}\right] \qquad \text{(A11.14)}$$

where

$$a = \frac{1}{2}(\beta_1+\beta_2+\beta_3)$$

$$b = \beta_1(V_{T1}-V_{Tij})+\beta_2(V_{T2}-V_{\theta i})+\beta_3(V_{T3}-V_{i+})$$

$$c = \beta_1((V_{Tij}-V_{T1}-\frac{V_j}{2})V_j)+\beta_2((V_{\theta i}-V_{T2}-\frac{V_{DD}}{2})V_{DD})$$

Figure A11.5 shows that the results produced by Equation A11.14 correspond very well to the values given by HSPICE Level 2 simulations. The negative sign of the gain values is important as the use of the positive input terminal of the operational amplifier

necessitates a negative feedback loop gain for the system to be stable. Over the chosen weight range, as the magnitude of the feedback loop gain is always less than 0.6, the system is stable. The changes in the gain with the weight voltage are related to the magnitude of the current in transistor M1, I_{M1}. For small values of I_{M1} the proportion of current at the common node due to the feedback transistor M3 is large. Thus a change in V_{i+} has a relatively large effective on V_{Ref}. As V_{Tij} and V_j increase, I_{M1} also increases. So the proportion of the current at the common node due to M3 decreases, resulting in V_{i+} having a smaller influence on the changes in V_{Ref}, thus accounting for the decrease in the magnitude of the gain as both V_{Tij} and V_j increase.

With the stability of the synapse circuit now proven, the feedback operational amplifier can be designed. One of the problems with large numbers of on-chip operational amplifiers is that of high power consumption. To make the power consumption more manageable, the load transistors (M9 and M10 in Figure 4.12), which normally determine the tail current have been replaced by a current mirror whose current is determined externally (Figure A11.6). The tail current is then mirrored to either the feedback amplifiers in the neuron or V_j buffer amplifiers. Unfortunately the operational amplifiers are now linked by a common current mirror, increasing the coupling of signals between operational amplifiers. In this case, in the absence of integrators, and with the feedforward nature of the vector matrix multiplication, the system simply takes longer to settle to its final voltage, the value of which is unchanged.

The specifications for the feedback and the V_j buffer operational amplifiers were different, in that the feedback amplifier was required to buffer the V_i voltage off-chip through an analogue output pad. Thus the feedback amplifier was compensated for loads up to 30pF while the buffer amplifier was designed to drive up to 20pF of load. To maintain either V_j or V_{Ref} to 1% accuracy a gain of at least 1000 was required. HSPICE simulations showed that if the system was to settle in under $10\mu s$. the slew rates of the operational amplifiers had to be greater than $5V/\mu s$

Cell	Size
Double Synapse	$200\mu m \times 115\mu m$
Feedback Op-amp	$200\mu m \times 240\mu m$
V_j Buffer Op-amp	$260\mu m \times 115\mu m$
Subtractor	$200\mu m \times 100\mu m$

Table A11.2 Cell Sizes for Synapse and Neuron Circuits.

A11.3 Layout

The synapse and operational amplifiers plus support circuitry were laid out using the Magic custom layout tool for ES2's 2 μm digital process. The resulting cell sizes are show in Table A11.2. To yield a more compact implementation two synapses were laid out as one cell. This reduces the number of well crossings required and thus the area of the cell. While the basic circuits of the V_j buffer amplifiers and the main feedback amplifiers were the same, the layouts were different. To maintain maximum accuracy in the neural multiplication and summation the differential inputs (M1, M2 Figure A11.6) for the feedback amplifiers in the neuron were cross-coupled with individual N-wells to eliminate body effect problems. Neither of these layout techniques is used in the V_j buffer amplifier, so that it was small enough to be pitch-matched to the synaptic array.

An 8 by 4 array of these synapses and the associated feedback amplifiers was then created for inclusion in a testchip. Alongside this array on the testchip were 2 other arrays based on variations of the pulse width modulation technique designed by Stephen Churcher [7] and Jon Tombs [8] respectively. ES2's Solo 1400 software was used to add pads and decoder logic to these arrays to complete the testchip. Unfortunately due to insurmountable funding problems this testchip was not fabricated.

A11.4 Conclusions

The virtues of this continuous time neural system are that is fast, cascadable, highly linear and nominally process invariant. Ultimately, the speed of the system depends on the performance of the amplifiers used. The amplifiers used here give a settling time of 5-10μs but with faster amplifiers the system should settle in under 1μs. As the testchip was to have been the precursor to a much larger implementation, the slower amplifier settling time offered a more appropriate compromise between speed, power consumption and area. Also to provide tight control over the operational amplifier's power consumption the tail current for the amplifier was determined externally.

While the HSPICE simulation results indicate that the system's performance will only vary by ±2% due to variations in the process, without results from fabricated circuits it is not possible to say what the true level of process invariance is.

A potentially serious limitation of the system is the absence of a sigmoid function to determine the neuron's neural state. It proved impossible to develop circuitry for a sigmoidal voltage to voltage transfer function that was both process invariant and continuous time in nature. A sigmoidal transfer characteristic is not required for the Kohonen Self-Organising network and thus it would be a suitable network to implement using this system. Other possible applications include analogue correlators and filters, or any algorithm which is based on a multiply accumulate architecture.

References

1. P.B. Denyer and J. Mavor, "Monolithic 256-point Programmable Transversal Filter", *Electronic Letters*, vol. 15, no. 22, pp. 710-712, October, 1979.
2. P.B. Denyer, "Design of Monolithic Programmable Transversal Filters using Charge-Coupled Device Technology", *PhD. Thesis (University of Edinburgh, UK)*, 1980.
3. P.B. Denyer and J. Mavor, "MOST Transconductance Multipliers for Array Applications", *IEE Proc. Pt. 1*, vol. 128, no. 3, pp. 81-86, June, 1981.
4. Il S. Han and Song B. Park, "Voltage-Controlled Linear Resistors by MOS Transistors and their Application to Active RC Filter MOS Integration", *Proc. IEEE*, pp. 1655-1657, November, 1984.
5. P.J. Ryan, D.G. Haigh, M. Banu, and Y. Tsividis, "Fully Integrated Active RC Filters in MOS Technology", *IEEE Journal of Solid-State Circuits*, vol. SC-18, no. 6, pp. 644-651, December, 1983.
6. M. Ismail, S.V. Smith, and R.G. Beale, "A New MOSFET-C Universal Filter Structure for VLSI", *IEEE Journal of Solid-State Circuits*, vol. SC-23, no. 1, pp. 183-194, February, 1988.
7. S. Churcher, "VLSI Neural Networks for Computer Vision", *PhD. Thesis (University of Edinburgh, UK)*, 1993.
8. J. Tombs, "MLP NN's and their Implementation in Analogue VLSI", *PhD. Thesis (Oxford University, UK)*, 1992.

Appendix 12

Published Papers

The follwing is a list of the papers published during the course of this PhD. The papers have not been included to keep the size of the thesis down, but the papers are available if required.

A. F. Murray, D. Baxter, Z. Butler, S. Churcher, A. Hamilton, H. M. Reekie and L . Tarassenko "Innovations in Pulse Stream Neural VLSI : Arithmetic and Communications" *IEEE Workshop on Microelectronics for Neural Networks*, Dortmund, pp. 8-15, 1990.

A. F. Murray, L. Tarassenko, H. M. Reekie, A. Hamilton, M. Brownlow, D. Baxter and S. Churcher "Pulsed Silicon Neural Nets - Following the Biological Leader" in *Introduction to VLSI Design of Neural Networks*, U. Ramacher, Kluwer, pp. 10 3-123, 1991.

D. J. Baxter, A. F. Murray and H. M. Reekie "Fully Cascadable Analogue Synapses Using Distributed Feedback" *Proc. Int. Workshop on VLSI for Artificial Intelligence and Neural Networks*, F4/ 1-F4/9, September 1990.

D. J. Baxter, A. F. Murray and H. M. Reekie "Fully Cascadable Analogue Synapses Using Distributed Feedback" in *VLSI for Artificial Intelligence and Neural Networks*, J. G. Delgado-Frias, W. R. Moore, Kluwer, pp. 205-213, 1991.

D. Baxter, A. F. Murray, H. M. Reekie, S. Churcher and A. Hamilton "Analogue CMOS Techniques for VLSI Neural Networks : Process-Invariant Circuits and Devices" *Proc. IEE Electronics Division Colloquium on Advances in Analogue VLSI*, May 1991.

D. Baxter, A. F. Murray, H. M. Reekie, S. Churcher and A. Hamilton "Advances in the Implementation of Pulse-Stream Neural Networks" *European Conference on Circuit Theory and Design 91*, Vol II, pp. 422-430 September 1991. (Invited Paper).

S. Churcher, D. J. Baxter, A. Hamilton, A. F. Murray and H. M. Reekie "Towards a Generic Neurocomputing Architecture" *International Conference on Neural Networks*, Munich, October 1991.

A. Hamilton, A. F. Murray, D. J. Baxter, S. Churcher, H. M. Reekie and L. Tarassenko "Integrated Pulse-Stream Neural Networks - Results, Issues and Pointers" *IEEE Trans. Neural Networks*, Vol 3, No. 3, May 1992.

S. Churcher, D. J. Baxter, A. Hamilton, A. F. Murray and H. M. Reekie "Generic Analog Neural Computation - The EPSILON Chip" *Advances in Neural Information Processing*

Systems 5, 1993.